Proyecto fin de Máster
Máster Universitario en Ingeniería Industrial

Tecnología Blockchain y su aplicación en la Automática

Autor: Carlos Bordons Martínez

Tutor: Luis Fernando Castaño Castaño

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Ingeniería de Sistemas y Automática

Proyecto fin de Máster
Máster Universitario en Ingeniería Industrial

Tecnología Blockchain y su aplicación en la Automática

Autor:
Carlos Bordons Martínez

Tutor:
Luis Fernando Castaño Castaño
Dr. Ingeniero Industrial

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Proyecto fin de Máster: Tecnología Blockchain y su aplicación en la Automática

Autor: Carlos Bordons Martínez
Tutor: Luis Fernando Castaño Castaño

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Quisiera aprovechar estas líneas para agradecer a todas las personas que han hecho posible la realización de este Trabajo Fin de Máster que marca el final de mi etapa como estudiante de la Universidad de Sevilla.

En especial me gustaría agradecer a mi familia, mis padres Carlos y Ana y a mi hermana Marta que me han empujado a terminar a tiempo el proyecto, a mi pareja Laura que ha sufrido junto a mi muchas horas de investigación y experimentos, a mis amigos y compañeros que me han apoyado durante todos estos meses y me han empujado a no distraerme y a mi tutor Fernando, sin cuya ayuda este proyecto no habría salido adelante en tiempo y forma, también quiero agradecer a la comunidad Blockchain de Boston que me ofrecieron la oportunidad de conocer este nuevo mundo tecnológico.

Carlos Bordons Martínez

Sevilla, 2019

Resumen

Este proyecto consiste en la introducción y descripción de la tecnología Blockchain, su importancia y su implementación en un sistema real.

En este caso, el sistema consistirá en un control de temperatura con dispositivos IoT controlado por Blockchain.

La implementación del bucle de control PI en el sistema descrito radica en el desarrollo y ejecución del controlador en un Smart Contract en la Blockchain.

La primera parte del proyecto consta de una introducción a la tecnología Blockchain, sus principales características, por qué es importante, qué es y cómo funciona.

La segunda parte describe como aplicar esta tecnología al campo del control automático.

La tercera parte consiste en un experimento de control de temperatura implementado en tecnología Blockchain.

Abstract

This project consists of the introduction and description of Blockchain technology, its importance and its implementation in a real system.

In this case, the system will consist of a temperature control with IoT devices controlled by Blockchain.

The implementation of the PI control loop in the described system lies in the development and execution of the controller in a Smart Contract in the Blockchain.

The first part of the project consists of an introduction to Blockchain technology, its main characteristics, why it is important, what it is and how it works.

The second part describes how to apply this technology to the field of automatic control.

The third part consists of a temperature control experiment implemented in Blockchain technology.

Índice

Índice de Figuras

Índice de Tablas

1 Introducción

1.1 Contexto

1.1.1 La importancia de la confianza

La sociedad se encuentra actualmente viviendo la Revolución Digital o Tercera Revolución Industrial, que marcó el comienzo de la Era de la Información.(Bryant, 2011)

La llegada de Internet permite a cualquier persona o máquina estar conectado con el resto del mundo, la privacidad y protección de datos se han convertido en una prioridad, la posibilidad de manejar información sensible se encuentra focalizada en pocos individuos y esto abre la posibilidad de la existencia de brechas de seguridad o uso fraudulento de esa información, la sociedad vive en la Era de la Información pero sin tener control sobre esa información (Romero, 2007).

Actualmente existen una decena de empresas/entidades que manejan la mayoría del tráfico del mundo y que actúan como intermediarios de los datos, son las "**Trusted Third Parties**" o Terceros de confianza, su cometido es gestionar los datos entre dos o más agentes, y se deposita en ellos la confianza de que las transacciones, acuerdos o trabajos a través de Internet entre estos agentes se llevarán a cabo (Cantor, 2018), algunos ejemplos serían:

- **Facebook:** Gestiona datos personales de un usuario con el resto de usuarios de la red, con la capacidad incluso de censurarlos o usarlos para otro fin.
- **PayPal:** Gestiona los pagos entre dos usuarios, con capacidad de bloquear o retirar el pago en caso de conflicto.
- **AirBnB/Uber:** Gestionan alquileres de inmuebles y servicios entre usuarios de su red.
- **Bancos y compañías de tarjetas de crédito:** Hacen de intermediario entre las cuentas bancarias de comprador y vendedor.

Todas estas plataformas crean su valor como gestores de datos donde los usuarios depositan su confianza. La tecnología Blockchain aparece para suplir la necesidad de acudir a terceros de confianza para realizar transacciones en Internet con plena **confianza** de que se llevarán a cabo, al igual que no es necesario un notario para realizar la compra del mes o pedir un taxi.

1.1.2 Historia de la tecnología Blockchain

En los años 90 un grupo de personas comenzó a reunirse para discutir problemas de programación y criptografía, esas reuniones evolucionaron a través de una lista de emails, a discusiones sobre matemáticas, informática, privacidad, libertad, filosofía y política, el grupo creció y terminaron haciéndose llamar "Cypherpunks" (imchris, 2004; PetriB, 2018).

Un Cypherpunk es cualquier activista que aboga por el uso generalizado de criptografía y tecnologías que mejoren la privacidad como ruta hacia el cambio social y político (PetriB, 2018; Manne, 2011).

En 1993 apareció un movimiento siguiendo el **"Cypherpunk manifesto**" cuya clave principal era la privacidad. Basado en ese principio se desarrollaron las primeras ideas sobre las divisas digitales (PetriB, 2018; Manne, 2011; Narayanan, 2013).

Existieron varios intentos de aplicar transacciones anónimas y mejoras en la privacidad en la red:

- **Adam Back (1997):** Creador de "Hascash", un mecanismo anti-spam que añade tiempo computacional al envío de emails, lo que convierte el envío de spam en un proceso costoso (Back, 2002).
- **Wei Dai (1997-1998):** Publicó la propuesta de "B-Money", un sistema anónimo de dinero efectivo electrónico distribuido, un método para almacenar la información de las transacciones sin ser necesaria la existencia de Third Parties (Dai, 1998).
- **Hal Finney (2004):** Creador del "Reusable Proof of Work", un software que soluciona el problema del "doble gasto", problema fundamental de las divisas digitales (nakamotoinstitute, 2004).
- **Nick Szabo (2005):** Publicó una propuesta para "Bitgold", un intento de replicar las propiedades económicas del oro mejorando sus propiedades de seguridad, los beneficios principales eran la independencia de instituciones financieras y transacciones entre países sin problemas (Moskov, 2018).

El primer Blockchain

El primer trabajo en una cadena de bloques asegurados criptográficamente fue descrito en 1991 por Stuart Haber y W. Scott Stornetta (Narayanan y col., 2016; Haber y Stornetta, 1990). Querían implementar un sistema donde las marcas de tiempo de los documentos no pudieran ser alteradas. En 1992, Bayer, Haber y Stornetta incorporaron árboles Merkle al diseño, lo que mejoró su eficiencia al permitir que varios certificados de documentos se reunieran en un solo bloque (Narayanan y col., 2016; Bayer, Haber y Stornetta, 1993).

El primer Blockchain fue conceptualizado por una persona (o grupo de personas) conocida como **Satoshi Nakamoto** en 2008. Nakamoto mejoró el diseño de una manera importante, utilizando un método similar al de Hashcash para agregar bloques a la cadena sin requerir que estén firmados por un "Trusted Party" (Narayanan y col., 2016). El diseño fue implementado el año siguiente por Nakamoto como un componente central de la criptomoneda **bitcoin**, que sirve como el registro público para todas las transacciones en la red (Economist, 2015).

1.1.3 El Bitcoin

Conceptos básicos:

- **Divisas:** La moneda o unidad monetaria es una unidad de cambio que facilita la transferencia de bienes y servicios, elimina la necesidad del trueque, la adopción de divisas puede ser global, nacional o local.

- **Bancos:** Entidades que almacenan las divisas de los usuarios, proporcionando una promesa de pago (extracto bancario) que da derecho a retirar esa cantidad almacenada, el extracto bancario refleja las transacciones de divisas del usuario con el banco. A cambio de pequeñas comisiones, el banco ofrece el servicio de ponerse en contacto con bancos de otros usuarios para permitir transacciones entre usuarios de distintas entidades, el servicio consiste en actualizar los extractos bancarios de ambos usuarios, evitando el "doble gasto" o que un usuario utilice el mismo dinero dos veces.

  Desventajas de este sistema bancario:

  1. Transferencias internacionales muy costosas.
  2. Es un sistema lento, la verificación y los servicios de transferencia pueden tardar varios días en completarse.
  3. Las cuentas pueden ser congeladas, o su saldo parcial o totalmente confiscado.
  4. Los bancos y otros procesadores de pagos como PayPal, Visa y Mastercard pueden negarse a procesar pagos para ciertas entidades legales.
  5. Es necesario tener confianza plena en las Trusted Parties (bitcoin.it, 2011b).

- **Reserva fraccionaria:** es un sistema bancario en el cual éstos mantienen sólo una fracción del monto de los depósitos de sus clientes como reserva, teniendo al mismo tiempo la obligación de retornar esos depósitos en demanda, es decir, en nuevos préstamos para sus clientes.
  Este sistema está basado en el hecho de que los depositantes no suelen reclamar todos sus depósitos al mismo tiempo, ni tampoco todos los prestamistas prestan al mismo tiempo, ni todos los deudores pagan al mismo tiempo. Dado su funcionamiento, el sistema expande la cantidad de dinero en circulación (este fenómeno se denomina el **multiplicador bancario**).
  En consecuencia, y dada la prevalencia del sistema, el agregado monetario de un país es generalmente mayor que la base monetaria.
  Economistas, notablemente los partidarios de la escuela austriaca, argumentan que el principal negocio de la banca de reserva fraccional es la creación de dinero, generalmente criticando este aspecto, dado que consideran que, debido a un privilegio injustificable otorgado por el estado a la banca, esta causa **deuda e inflación** (Golin y Delhaise, 2013; Cochran y Call, 2000; Capella, 2009).

Satoshi Nakamoto en su paper "**Bitcoin: A Peer-to-Peer Electronic Cash System**" (Nakamoto y col., 2008) crea un sistema que permite realizar transacciones globales, seguras, rápidas y de muy bajo coste, en la que el doble doble gasto de un activo digital no es posible, sin depender de terceros agentes como bancos o procesadores de pagos, se trata de

un sistema que defiende la banca de reserva 100 %, donde no se puede crear dinero con el método de la reserva fraccionaria, evitando la devaluación del activo con el tiempo.

El 3 de enero de 2009, la red de bitcoins se creó cuando Nakamoto generó el primer bloque de la cadena, conocido como el bloque de génesis (**Genesis Block**).

Incluido en la coinbase de este bloque está el siguiente texto: "**The Times 03/Jan/2009 Chancellor on brink of second bailout for banks**" (El Times 03/Ene/2009 Canciller a punto de segundo rescate para los bancos). Esta nota se ha interpretado como una marca de tiempo y un comentario sobre la inestabilidad causada por la banca de reserva fraccionaria (oroyfinanzas, 2015; Davis, 2011).

1.2 Motivación

Durante 2018-2019 he realizado una estancia en Boston, USA, dónde he tenido contacto con esta tecnología y cuando se estudia con calma y en profundidad se da uno cuenta de las posibilidades que el **Blockchain** tiene para mejorar la interacción de las personas y las máquinas con Internet, y lo necesario que van a ser sus funcionalidades con la llegada del Internet de las Cosas (IoT) y del Cloud Computing en términos de **privacidad, seguridad y confianza**.

Este conocimiento, unido a mi formación en ingeniería automática me ha motivado a realizar este proyecto e intentar unir ambas disciplinas.

Actualmente el Blockchain solo se conoce por su potencial en criptodivisas, pero a lo largo de este Trabajo de Fin de Máster quiero mostrar su potencial para integrarse con las demás tecnologías, empresas y en particular con la rama del control automático.

1.3 Objetivos y alcance del proyecto

El principal objetivo del presente proyecto es estudiar si existe campo de mejora en el sector del control automático al aplicar la tecnología Blockchain y estudiar cómo esta tecnología novedosa puede ayudar a integrar el control automático con Internet (descentralización, seguridad, computación en la nube) y mejorar ciertos detalles de la implementación actual del control automático en la industria.

2 ¿Por qué el Blockchain?

En el capítulo 1, se ha destacado la importancia que tienen la **confianza** y la **seguridad** en nuestra sociedad, y como ambos atributos dependen en la actualidad de terceros agentes los Trusted Third Parties. En este capítulo haremos hincapié en como el Blockchain puede mejorar el funcionamiento de estos agentes o incluso sustituirlos.

Las principales desventajas de las transacciones en la red son que son **lentas** y dependen de **intermediarios** que normalmente añaden un **coste** a las transacciones por sus servicios, a cambio se aseguran de que estas transacciones sean **seguras**, sean **verdaderas/**no fraudulentas y **fiables**.

El objetivo de la tecnología Blockchain es por tanto conseguir transacciones seguras, confiables y verdaderas entre los usuarios, rápidas y a bajo coste, sin necesidad de intermediarios con acceso y control sobre los datos de los usuarios de la red.

Con Blockchain, podemos imaginar un mundo en el que los contratos se incorporan en el código digital y se almacenan en bases de datos transparentes y compartidas, donde están protegidos contra la eliminación, la manipulación y la revisión. En este mundo, cada acuerdo, cada proceso, cada tarea y cada pago tendrían un registro y una firma digital que podrían identificarse, validarse, almacenarse y compartirse. Los intermediarios como abogados, corredores y banqueros ya no serían necesarios. Los individuos, las organizaciones, las máquinas y los algoritmos transarían e interactuarían libremente entre sí con poca fricción. Este es el inmenso potencial de blockchain (Lansiti, 2017).

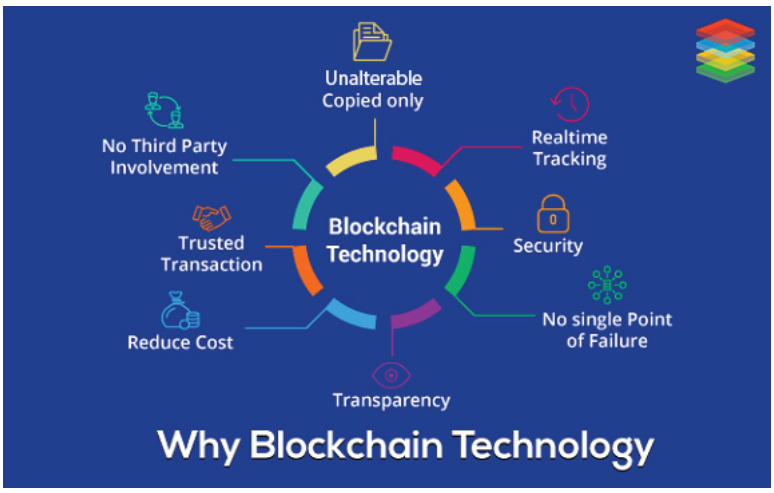


Figura 2.1 ¿Por qué el Blockchain?(Gill, 2018).

2.0.1 El Internet del valor

El "**Internet del valor**" representa un mundo donde el **dinero** y los **activos** se intercambian a la misma velocidad a la que la información se mueve hoy. Las transacciones se producen en tiempo real y en redes globales, resolviendo el problema de los sistemas de pago internacionales que no son interoperables o que los activos no se pueden trasladar entre fronteras.

El problema que tenemos hoy en el mundo es que las redes de pago, incluso cuando son eficientes, son no interoperables. Por primera vez, existe una tecnología que resuelve ese problema de falta de interoperabilidad entre redes y permitirá que el **valor** se mueva como **información** (RSK educate, 2015; CME Open Markets - CME Group, 2015).

2.1 La verdad online, actual vs Blockchain

Conceptos:

- **Ledger:** Cuya traducción directa es libro mayor o libro de cuentas, pero que en términos de Blockchain abarcará un ámbito mayor como registro de transacciones (normalmente una base de datos), los Ledgers se llevan usando desde el 3000 a.C (En Wikipedia, s.f.[e]).
- **Nodo:** En redes de computadoras cada una de las máquinas es un nodo, y si la red es Internet, cada servidor constituye también un nodo (En Wikipedia, s.f.[r]).

Actualmente todas las transacciones en los distintos nodos de la red quedan registradas, ese registro es lo que se va a denominar “Ledger” para la comparación de las tecnologías. Por tanto todas las transacciones que ocurren en la red quedan registradas en el Ledger de cada nodo principal, si una transacción no está en el Ledger, a efectos reales no ha ocurrido, por tanto ese **Ledger es “La verdad”** de esa red.

Es muy importante definir **quién** puede escribir y editar ese Ledger:

- **Internet y comunicaciones actuales:** el Ledger es escrito por uno o varios nodos principales, y se encuentra centralizado en un nodo o descentralizado en varios nodos para aumentar la seguridad y evitar que el nodo principal se convierta en un

"Honeypot" para los hackers que intenten alterar "La verdad" de esa red en su beneficio, o robar la información sensible almacenada (En areatecnologia, s.f.[b]; En areatecnologia, s.f.[a]).

- **Blockchain:** el Ledger está **distribuido**, todos los nodos de la red pueden tener permiso para escribir si cumplen unos requisitos y la propia tecnología impide que se alteren registros antiguos.
  Al estar completamente distribuido entre todos los nodos, en caso de alterarse el Ledger de un nodo de manera fraudulenta, comparando con las copias del resto de la red es fácil discernir el Ledger manipulado del Ledger "global", que contiene las verdaderas transacciones, sería necesario modificar el 51 % de los nodos para alterar "La verdad" de esa red, por tanto la resistencia a hackers aumenta con el aumento del número de nodos que conforman la red (En Wikipedia, s.f.[c]).

Figura 2.2 Tipos de redes (Rautopia, s.f.).

Me gustaría compartir la siguiente cita del libro **"The Truth Machine":**

Puede que te sorprenda leer esto, pero la idea más subversiva, controvertida y antiautoritaria en el mundo de las finanzas, una idea tan poderosa que todos los gobiernos del planeta está tratando de averiguar si la utilizan o la prohíben, el sueño de los libertarios más fervientes y los habitantes de la dark-web, es un **"Ledger"** (Vigna y Casey, 2019).

Como veremos más adelante el resultado de la tecnología Blockchain es un método de registro que nos lleva a una versión comúnmente aceptada de la **verdad** que es más **confiable** que cualquier otra verdad que hayamos visto. Estamos llamando a la cadena de bloques una Máquina de la Verdad, y sus aplicaciones van mucho más allá del dinero (Vigna y Casey, 2019).

2.2 La seguridad online, actual vs Blockchain

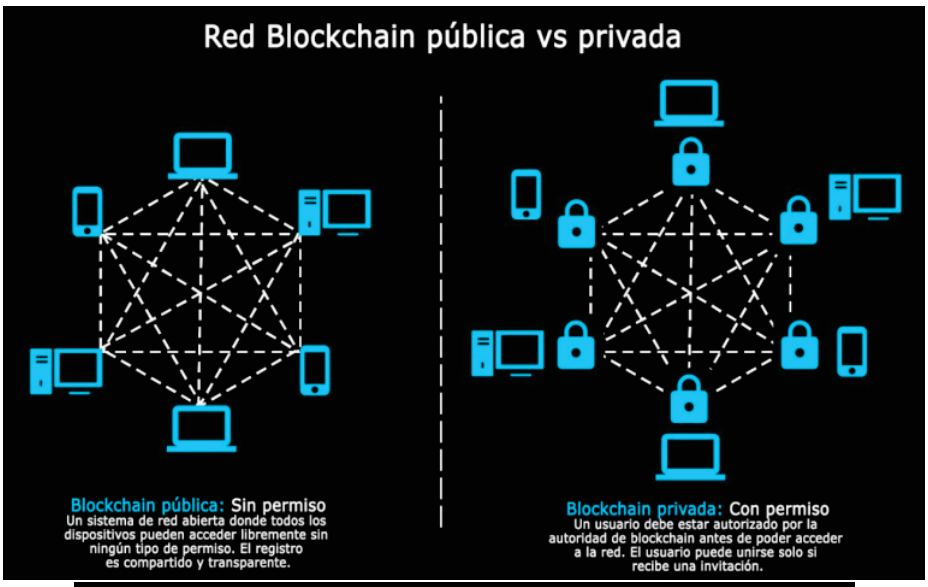
La seguridad online es el reto de la actual generación de criptógrafos y matemáticos, es primordial que el Ledger o las bases de datos, no sean modificados con fines fraudulentos por terceros ajenos a la red o por suplantadores de identidad.

2.2.1 Tipos de Redes:

- **Internet y comunicaciones actuales:** normalmente son redes privadas donde es necesario un **usuario y contraseña** para acceder o híbridas que permiten acceder a parte de la red sin identificarse, Internet puede considerarse una gran red pública con servidores públicos y/o privados (En areatecnologia, s.f.[a]).

- **Blockchain:** existen 4 tipos de Blockchain (Preukschat, 2017):

    1. **Blockchains públicas:** son accesibles a cualquier usuario en el mundo. Lo único que se necesita es un ordenador y una conexión a Internet (ej: Bitcoin, Ethereum).
    2. **Blockchains privadas:** no está abierta al público, sino que solo se puede acceder a ella por invitación, normalmente controlado por una sola organización (ej: Hyperledger).
    3. **Blockchains de consorcios:** se dice que es semi-descentralizado. También el acceso está autorizado, pero en lugar de que una sola organización lo controle, varias compañías pueden operar un nodo en dicha red (ej: Quorum).
    4. **Blockchains híbridas:** Las Blockchain híbridas son una combinación de las públicas y privadas. En una Blockchain híbrida los nodos participantes son invitados, pero todas las transacciones son públicas. Eso quiere decir que los nodos participan en el mantenimiento y seguridad de esta Blockchain, pero todas las transacciones son visibles para usuarios en todo el mundo, a diferencia de las Blockchains privadas en la cual las transacciones son privadas también (ej: Evernym).



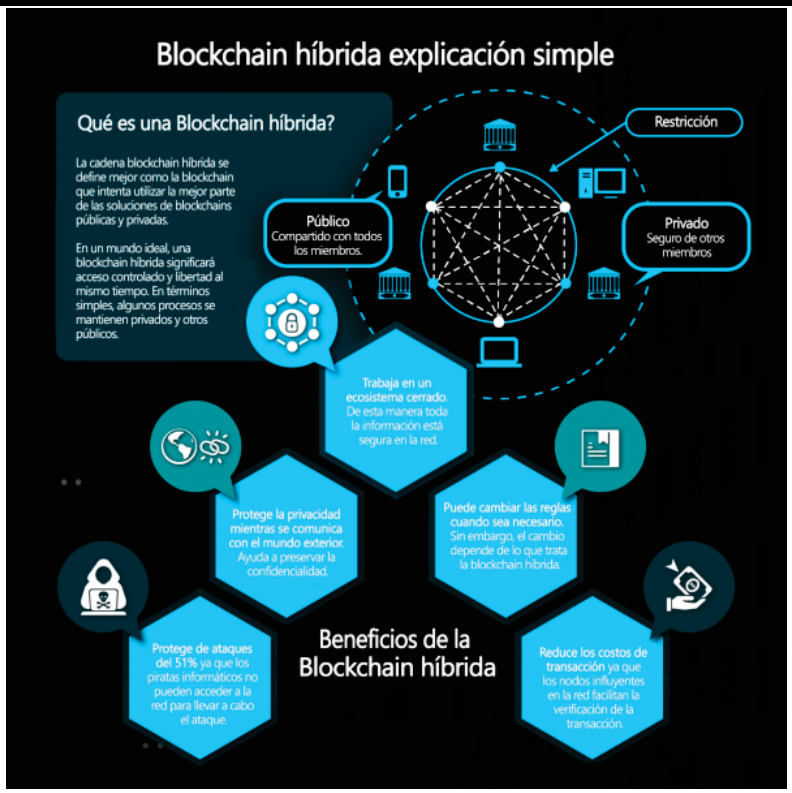


Figura 2.3 BC Pública vs Privada vs Híbrida (Pérez, 2019).

2.2.2 Interacción con la red:

- **Internet y comunicaciones actuales:** para interactuar con los servidores de una red primero debes acceder a ella y segundo tener permiso para interactuar, normalmente ambas acciones se realizan con la combinación de usuario y contraseña, que suele ser uno de los puntos más **vulnerables** de la red.
  Las redes privadas normalmente están protegidas por diferentes métodos de seguridad como VPNs, Firewalls y protocolos de seguridad y encriptación (Tactical Technology Collective y Front Line, 2018).

- **Blockchain:** el punto clave de la seguridad en la tecnología Blockchain es que utiliza la tecnología de **Llave Pública-Llave Privada** (Public-Private key pair), este método de encriptación elimina la necesidad de verificar el dúo usuario y contraseña contra una base de datos.
  La seguridad para acceder a la red se garantizará siempre que la Llave Privada de cada usuario se mantenga protegida o sea cambiada regularmente.
  Para Blockchains privadas se pueden utilizar los mismos métodos de seguridad ya mencionados (VPNs, Firewalls, etc) (Joshi, Han e Y. Wang, 2018).

La gran ventaja del método de las claves públicas-privadas de las Blockchains contra el dúo de usuario y contraseña, es que en caso de desvelarse la clave privada de un usuario, solo las transacciones/acciones de ese usuario estarían comprometidas y sería necesario conocer todas las claves privadas, que **NO se almacenan** en ninguna base de datos, para poder acceder a la información del resto usuarios.

En casos de acceso fraudulento a una base de datos de un servidor que contenga todos los usuarios y contraseñas y demás información sensible (por ejemplo con el usuario y contraseña del Administrador), si la información no está correctamente cifrada o son contraseñas débiles, puede afectar a toda la red de usuarios de ese servidor, como ya ha pasado en vulnerabilidades pasadas de algunas empresas (En Wikipedia, s.f.[s]).

En el caso de la tecnología Blockchain, si se desvelase la clave Privada del Administrador, el resto de claves Privadas permanecen seguras ya que no se encuentran almacenadas en el Servidor/Base de datos comprometido.

2.3 La confianza online, actual vs Blockchain

2.3.1 ¿Qué es la confianza?

Esperanza firme que una persona tiene en que algo suceda, sea o funcione de una forma **determinada**, o en que otra persona actúe como ella desea (En Lexico, s.f.).

Fuera de la red esta confianza está proporcionada por figuras como los notarios, que se ocupan de registrar las transacciones como válidas, los cuerpos del estado como el judicial y las fuerzas de seguridad que se encargan de que los contratos y acuerdos lleguen a cumplirse asegurando la confianza de los acuerdos y en última instancia el dinero en efectivo que al hacer un pago en mano provee la seguridad de que la persona recibirá en ese mismo instante el objeto o servicio adquirido Vigna y Casey (2019).

2.3.2 Confianza Online

La falta de confianza ha sido identificada repetidamente como una de las **barreras** más formidables para que las personas se involucren en el comercio electrónico, involucrando transacciones en las cuales la información financiera y personal se envía a los comerciantes a través de Internet.

El futuro del comercio electrónico es tenue sin un clima general de confianza en línea. **Crear confianza** en los consumidores en Internet representa un desafío para los comerciantes en línea y es un tema de investigación de creciente interés e importancia (Y. D. Wang y Emurian, 2005).

- **Internet y comunicaciones actuales:** como se ha comentado en el Capítulo 1, en la sociedad actual la confianza online la proveen los Third Trusted Parties. Cuando se realiza un pago online se utiliza PayPal, que hace de intermediario entre el comerciante y el usuario, cuando se compra un articulo online se usan páginas como Amazon que proveen la confianza de que el articulo será entregado y en buenas condiciones y que los datos personales y financieros están seguros, para buscar información se accede a Wikipedia que asegura que la información que proveen es fiable, para la comunicación instantánea se utiliza Whatsapp o Facebook, intermediarios que aseguran que el mensaje llegará sin ser interceptado.
  Todos estos **intermediarios** tienen **acceso** a esta información sensible, pero se **confía** en que estos agentes no harán uso fraudulento y cumplirán con la utilización acordada de esos datos.

- **Blockchain:** la tecnología Blockchain pretende revolucionar la gestión de la confianza online, esta tecnología sustituye los Third Trusted Parties por los nodos.
  Los **nodos** serán los encargados de ejecutar el **código** que verificará que la transacción es correcta, una vez **validada** la transacción esta es añadida a la cadena de bloques asegurando que es verdad, y una vez añadida no puede ser modificada.
  La gran diferencia es que estos nodos no tienen acceso a los datos, esto significa que los **nodos no pueden modificar** o hacer un mal uso de los datos de una transacción firmada por una **clave Privada** y estas son solo propiedad del usuario, no pueden ser robadas en un ataque de hackers a un servidor.

La historia ya ha mostrado escándalos como el uso de datos fraudulento por Facebook o grandes webs hackeadas con datos de millones de usuarios filtrados, pero hasta ahora los usuarios no tenían otra alternativa que confiar en estos agentes, esto puede cambiar con la llegada de la tecnología Blockchain (R. Álvarez, 2018; Otero, 2018).

Un buen ejemplo de como funciona una **transacción** de un activo en la Blockchain sería el Bitcoin (bitcoin.it, 2011c):

1. El usuario A quiere enviar 1 BTC al usuario B, para ello crea la transacción y la **firma** con su **llave Privada**, después envía esa transacción firmada a la Blockchain Bitcoin.

2. La Blockchain (nodos) recibe la transacción firmada, comprueba que el usuario A tiene 1 BTC para enviar y si es correcto **valida** la transacción.

3. Se disminuye el saldo de A en 1 BTC y se aumenta el saldo de B en 1 BTC.

4. Una vez la transacción es validada se añade a la cadena de bloques de esta Blockchain y ya no puede ser **modificada**, evitando que A pueda volver a enviar ese BTC (problema del doble gasto).

5. Todo este proceso ocurre muy **rápido** o incluso instantáneo en otras Blockchains distintas al Bitcoin, con unos **costes muy bajos** por transacción, del orden de céntimos de euro o incluso menos sin importar la cantidad intercambiada.

En comparación con PayPal, un banco o la compra de una casa autenticada por un notario, en este ejemplo los nodos no pueden modificar el saldo de A sin su **llave privada**, ni congelarlo ni perderlo, potestad que sí tienen los agentes de confianza actuales.

Los BTCs del usuario A están seguros si un hacker accede a la base de datos de la Blockchain, ya que no puede modificar las cuentas/saldos ni puede acceder a las contraseñas porque las claves privadas no se almacenan en la Blockchain.

Un hacker que accediera con una cuenta de Administrador del sistema de cualquier Third Trusted Party actual, sí podría llegar a modificar transacciones de otros usuarios sin necesitar la clave de ese usuario, esto es imposible en la Blockchain (hashgains, 2018).

Por tanto los agentes de confianza actuales podrán ser sustituidos en el futuro por **nodos** ejecutando **código**, estos nodos son **seguros y confiables** debido a la robustez que proporciona la Blockchain **distribuida** que impide la existencia de nodos deshonestos que escriban transacciones falsas a su favor.

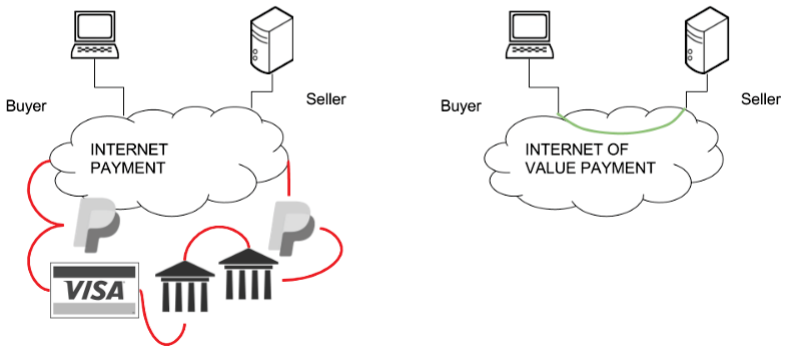


Figura 2.4 Multitud de intermediarios(Casals, 2016).

2.3.3 Smart Contracts frente a los proveedores online

Más adelante se estudiará en profundidad qué son los **Smart Contracts**, un **software** que permite ejecutar cualquier tipo de **código** en la **Blockchain**.

Esta tecnología permite realizar transacciones/acciones del tipo "si se cumple la condición A realiza B", e incluso permitirá ejecutar cualquier código con las propiedades clave de la tecnología Blockchain, **verdad**, **seguridad** y **confianza**, los Smart Contracts integrados con el Internet of Things tienen un gran potencial.

Un ejemplo sencillo puede ser un Smart Contract de viviendas de alquiler similar a AirBnB. **AirBnB**:

1. El usuario confía sus datos de pago a la plataforma, esta tendrá potestad de cargar cualquier cantidad al usuario, por ser una Third Trusted Parties es poco probable que haga mal uso de esos datos de pago.

2. El usuario debe contactar con el dueño de la vivienda y confiar en que el dueño cumplirá lo prometido, proveer el acceso a la vivienda.

3. La plataforma realizará el pago al dueño de la vivienda tras deducir una comisión por su papel como intermediario, la plataforma hará de árbitro si surge algún problema, puede incluso retener el pago o realizar cargos extra.

Un **Smart Contract** de viviendas de alquiler:

1. El usuario interactúa con el Smart Contract firmando el pago acordado con su clave privada.

2. El Smart Contract comprueba que se ha realizado el pago en la Blockchain y libera la información de acceso a la vivienda para el usuario y libera el pago para el dueño de la vivienda.

3. Se podrían añadir más condiciones al Smart Contract como verificación de la vivienda previo al pago o incluir un sistema de fianzas, incluso conectar la Blockchain con accesos domóticos a la vivienda para permitir/rescindir el acceso solo al usuario adecuado.

4. Por la ejecución del Smart Contract suele ser necesaria una pequeña comisión del orden de céntimos de euro.

La gran diferencia en este ejemplo es que en el futuro no será necesario una plataforma **intermediaria** que asegure el pago del servicio, o que ponga en contacto a las partes y que tenga el poder de **acceder** a los datos de pago o de **denegar** el acceso al servicio, además reduciendo las **comisiones** por realizar este servicio.

Es cierto que en este ejemplo se compara un Smart Contract con AirBnB, un agente de confianza en la sociedad actual.

Esta confianza la ha construido esta empresa a su vez creando un monopolio, donde evita que otras empresas con el mismo modelo de negocio puedan entrar a competir ya que no son igual de fiables. Estas empresas podrían ser fiables si utilizan tecnologías como los Smart Contracts, donde su negocio no fuera el de la confianza e intermediación, sino el de dar un buen servicio al usuario.

En una estructura de **Smart Contract**, la realización contractual requerida no depende de la confianza de las partes entre sí, o de un tercero, sino que dependen de la confianza de las partes en el propio Smart Contract, que no se basa en una **promesa de cumplimiento** sino en la **programación** diseñada para lograr un objetivo específico (Govender, 2019).

Figura 2.5 Contratos tradicionales vs Smart Contracts(W, 2018).

3 ¿Qué es el Blockchain?

En el Capítulo 2, se ha destacado el por qué la tecnología Blockchain es necesaria para mejorar la seguridad y confianza online y ahorrar costes de intermediación, permitiendo realizar transacciones verdaderas, fiables y seguras de forma rápida y a coste reducido.

En este capítulo 3 se procede a estudiar qué es la tecnología Blockchain y en el siguiente capítulo 4 se estudia cómo funciona esta tecnología para lograr estos objetivos tan ambiciosos.

3.0.1 Definición de Blockchain

Una Blockchain, es una **estructura de datos** en la que la información contenida se agrupa en conjuntos (**bloques**) a los que se les añade **información relativa** al bloque anterior de la cadena en una línea temporal, de manera que gracias a técnicas **criptográficas**, la información contenida en un bloque solo puede ser repudiada o editada modificando todos los bloques **posteriores,** esta Base de Datos está **distribuida** entre todos los nodos que participan en el sistema.

En la práctica ha permitido, gracias a la criptografía asimétrica y las funciones de resumen o hash, la implementación de un registro contable (ledger) **distribuido** que permite soportar y garantizar la seguridad del dinero digital.

Siguiendo un protocolo apropiado para todas las operaciones efectuadas sobre la Blockchain, es posible alcanzar un **consenso** sobre la integridad de sus datos por parte de todos los **participantes** de la red, sin necesidad de recurrir a una entidad de confianza que **centralice** la información.

Por ello se considera una tecnología en la que la "**verdad**" (estado confiable del sistema) es construida, alcanzada y fortalecida por los propios **miembros**; incluso en un entorno en el que exista una minoría de nodos en la red con comportamiento malicioso dado que, en teoría, para comprometer los datos, un atacante requeriría de una mayor potencia de cómputo y presencia en la red que el resultante de la suma de todos los restantes nodos combinados.

Por las razones anteriores, la tecnología Blockchain es especialmente adecuada para escenarios en los que se requiera **almacenar** de forma creciente datos ordenados en el tiempo, sin posibilidad de **modificación** ni revisión y cuya **confianza** pretenda ser **distribuida** en lugar de residir en una **entidad** certificadora (En Wikipedia, s.f.[d]).

Es un registro **único**, **consensuado** y **distribuido** en varios nodos de una red (Pastorino, 2018).

Una cadena de bloques es esencialmente solo un registro, un libro mayor de acontecimientos digitales que está "**distribuido**" o es compartido entre muchas partes diferentes. Solo puede ser actualizado a partir del **consenso** de la mayoría de participantes del sistema y, una vez introducida, la información nunca puede ser borrada. La cadena de bloques de Bitcoin contiene un registro certero y verificable de todas las transacciones que se han hecho en su historia (Bit2Me, 2018a).

3.1 Estructuras de Datos: Bases de Datos y Blockchain

3.1.1 Introducción a las Bases de Datos

Por qué utilizar una base de datos

Una base de datos (BD) proporciona a los usuarios el acceso a datos, que pueden visualizar, ingresar o actualizar, en concordancia con los derechos de acceso que se les hayan otorgado (Villagómez, 2017).

Qué es una base de datos

Una base de datos es una entidad en la cual se pueden almacenar datos de manera **estructurada**, con la menor redundancia posible.

Una base de datos se entenderá como una colección de datos **relacionados** entre sí y que tienen un **significado** implícito. Por datos se refiere a hechos conocidos que pueden registrarse y que tienen un significado implícito(Villagómez, 2017; readthedocs.io, s.f.).

Diferentes programas y diferentes usuarios deben poder utilizar estos datos. Por lo tanto, el concepto de base de datos generalmente está relacionado con el de red, ya que se debe poder **compartir** esta información.

Una base de datos puede ser local, es decir que puede utilizarla solo un usuario en un equipo, o puede ser **distribuida**, es decir que la información se almacena en equipos remotos y se puede acceder a ella a través de una red.

La principal ventaja de utilizar bases de datos es que **múltiples usuarios** pueden acceder a ellas al mismo tiempo (Villagómez, 2017).

Sistemas de ficheros tradicionales

En estos sistemas, cada programa almacenaba y utilizaba sus propios datos de forma un tanto **caótica**. La única ventaja que conlleva esto es que los procesos son **independientes**, por lo que la modificación de uno no afecta al resto (readthedocs.io, s.f.).

Pero tiene grandes inconvenientes:

- **Datos redundantes.** Ya que se repiten continuamente.
- **Coste de almacenamiento elevado.** Al almacenarse varias veces el mismo dato en distintas aplicaciones, se requiere más espacio en los discos.
- **Tiempos de procesamiento elevados.** Al no poder optimizar el espacio de almacenamiento.
- **Probabilidad alta de inconsistencia en los datos.** Esto se debe a que un proceso cambia sus datos y no el resto. Por lo que el mismo dato puede tener valores distintos según qué aplicación acceda a él.

- **Difícil modificación en los datos.** Debido a la probabilidad de inconsistencia, cada modificación se debe repetir en todas las copias del dato.

Sistemas de base de datos relacional

En este tipo de sistemas los datos se centralizan en una base de datos **común** a todas las aplicaciones(readthedocs.io, s.f.).

Sus ventajas son las siguientes:

- **Menor redundancia.** No es necesaria la repetición de datos. Aunque, sólo los buenos diseños de datos tienen poca redundancia.
- **Menor espacio de almacenamiento.** Debido a una mejor estructuración de los datos.
- **Acceso a los datos más eficiente.** La organización de los datos produce un resultado más óptimo en rendimiento.
- **Datos más documentados.** Los meta-datos permiten describir la información de la base de datos.
- **Independencia de los datos y los programas y procesos.** Esto permite modificar los datos sin modificar el código de las aplicaciones.
- **Integridad de los datos.** Mayor dificultad de perder los datos o de realizar incoherencias con ellos.
- **Mayor seguridad en los datos.** Al limitar el acceso a ciertos usuarios.

Como contrapartida encontramos los siguientes inconvenientes:

- **Instalación costosa.** El control y administración de bases de datos requiere de un software y hardware potente.
- **Requiere personal cualificado.** Debido a la dificultad de manejo de este tipo de sistemas.
- **Implantación más larga y difícil.**
- **La adaptación del personal es más complicada y lleva mayor tiempo.**

Sistemas de Gestión de Bases de Datos

Un sistema gestor de bases de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener una base de datos, y proporciona acceso controlado a la misma (readthedocs.io, s.f.).

En general, un SGBD proporciona los siguientes servicios:

- **Permite la definición de la base de datos mediante el lenguaje de definición de datos (DDL – Data Description Language).** Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos. Todo esto se almacenará en la base de datos.

- **Permite la inserción, actualización, eliminación y consulta de datos mediante el lenguaje de manejo o manipulación de datos (DML - Data Manipulation Language).**

- **Proporciona un acceso controlado a la base de datos** mediante:

  - Un sistema de seguridad, de modo que los usuarios no autorizados no puedan acceder a la base de datos, mediante el lenguaje de control de datos (DCL - Data Control Language);
  - Un sistema de integridad que mantiene la integridad y la consistencia de los datos;
  - Un sistema de control de concurrencia que permite el acceso compartido a la base de datos;
  - Un sistema de control de recuperación que restablece la base de datos después de que se produzca un fallo del hardware o del software;
  - Un diccionario de datos o catálogo accesible por el usuario que contiene la descripción de los datos de la base de datos.

3.1.2 Bases de Datos en la Blockchain

Centralizadas, descentralizadas y distribuidas

- **Centralizadas:** mantienen todos los datos en una **única** ubicación y para acceder a la información se debe ingresar al servidor principal del sistema.
  Su principal ventaja es que son más **sencillas** de mantener, pero son muy poco estables/resistentes a fallos, mayores **riesgos** de seguridad al contener toda la información en un solo lugar, poco resistentes a multitud de peticiones y problemas de **escalabilidad** (Rosa, 2018).

- **Descentralizadas:** consiste en una serie de servidores que se encuentran en **distintos** lugares geográficos, los datos no se almacenan en un solo lugar, sino que están almacenados en una serie de servidores distintos que proveen de información a los clientes.
  Funcionan como un grupo de bases de datos **independientes** sin conexiones lógicas entre ellas y que no están totalmente interconectadas mediante una red de comunicaciones. Son más **estables** que las centralizadas y poseen mayor **seguridad** al tener los datos almacenados repartidos y mayor **escalabilidad** (Rosa, 2018).

- **Distribuidas:** funciona como una **única** base de datos lógica que está instalada en una serie de servidores (nodos) ubicadas en **diferentes** lugares geográficos y que no están conectadas a una única unidad de procesamiento, pero si están totalmente **conectadas** entre sí a través de una red de comunicaciones.
  En este sistema todos los nodos contienen información y todos los clientes del sistema están en condición de **igualdad**. De esta forma las bases de datos distribuidas pueden realizar procesamientos **autónomos**. Son las más **estables** de todas al almacenar la totalidad de la información del sistema en un gran número de nodos que mantienen condiciones de igualdad entre sí, poseen la mayor **escalabilidad** de los 3 modelos (Rosa, 2018).

Cita de las palabras de **Vitalik Buterin**, creador de **Etherum** (Buterin, 2017):

Las Blockchains están **políticamente descentralizadas** (nadie las controla) y **arquitectónicamente descentralizadas** (no hay un punto de fallo central infraestructural) pero están **lógicamente centralizadas** (hay un estado comúnmente acordado y el sistema se comporta como una sola computadora).

¿Qué tipo de Base de Datos es una Blockchain?

Un Blockchain es una base de datos porque es un Ledger digital que almacena información en **estructuras de datos** llamadas bloques.
Una base de datos almacena información en estructuras de datos llamadas tablas.
Sin embargo, mientras que una cadena de bloques es una base de datos, una base de datos no es una cadena de bloques, aunque ambos almacenan información, difieren en el **diseño**.

Blockchain es un Ledger digital inmutable, es una base de datos distribuida en continuo crecimiento que está criptográficamente protegida.

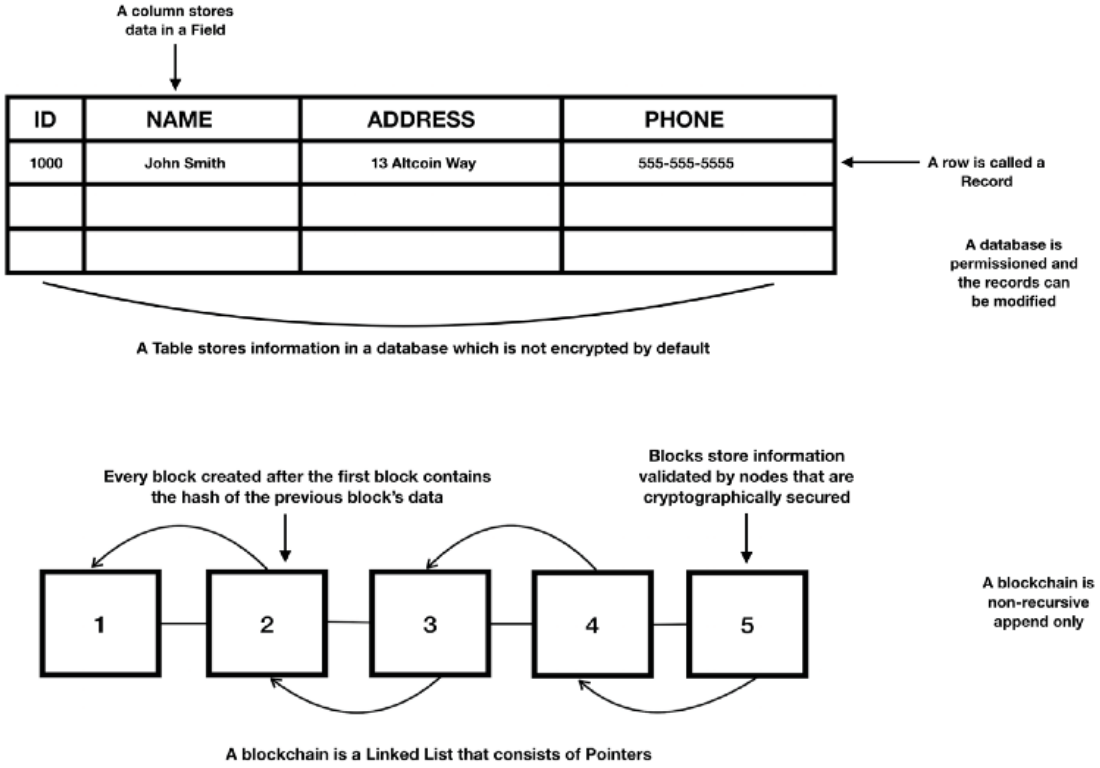


Figura 3.1 Diferencia estructuras de BD y Blockchain (Tabora, 2018).

Un Blockchain almacena información en bloques de tamaño **uniforme**. Cada bloque contiene la información de hash del bloque anterior para proporcionar seguridad **criptográfica**. Este hash contiene la información y la firma digital del bloque anterior, y de todos los hashes de los bloques anteriores que se remonta al primer bloque producido llamado "bloque **Génesis**". Una estructura de datos de Blockchain es un ejemplo de un árbol Merkle, que se utiliza como una forma eficiente de verificar los datos.

Para que los bloques se **agreguen** a una cadena de bloques, se utiliza un proceso basado en la teoría de juegos. Las computadoras que funcionan como nodos en la red llamadas "mineros" deben competir entre sí.

Una vez que un minero haya **validado** un bloque, recibirá una recompensa como incentivo por proporcionar sus recursos informáticos a la red y el bloque es añadido a la Blockchain.

La cadena de bloques utiliza una red **distribuida** de nodos que está **descentralizada**.

La descentralización significa que todos los nodos de la red almacenan una copia de la cadena de bloques. Los nodos almacenan una copia completa (nodos completos) y realizan operaciones de minería.

No existe un **administrador** para validar un bloque de transacciones.

Una vez que el bloque se ha agregado a la cadena de bloques, la información es **inmutable** y **transparente** para todos.

Las transacciones de Blockchain no son **recursivas**, lo que significa que no pueden repetirse una vez que se validen en un bloque.

Un Blockchain es altamente **tolerante a fallos** ya que si uno o más nodos están inactivos, siempre habrá otros nodos disponibles que ejecutarán el Blockchain.

Otra ventaja de la descentralización es que permite a las personas que no se conocen o no confían entre sí realizar transacciones. Lo que hace el Blockchain es proporcionar esa **confianza** a través de la transparencia al registrar la transacción y proporcionar una forma criptográfica **segura** de intercambiar valor (Tabora, 2018).

Bases de Datos vs Blockchain

A continuación se desgranan las diferencias entre Bases de Datos y las Blockchains, y como son compatibles en la sociedad actual dependiendo del uso que necesite el usuario para sus datos (Tabora, 2018).

Ventajas de las Bases de Datos:

- **Personalización para la facilidad de uso:** Las bases de datos centralizadas pueden ser personalizadas por el administrador dependiendo de los requisitos comerciales. También se puede distribuir a distintas ubicaciones en las que los datos se pueden combinar en una base de datos maestra para consultas e informes. Ofrecen características robustas que permiten a los desarrolladores crear aplicaciones para brindar a los usuarios una interfaz más consistente y fácil de usar.

- **Estabilidad:** Cuando se administra correctamente, un sistema de base de datos puede manejar grandes volúmenes de datos y procesar miles de transacciones por segundo. También son rápidos porque otorgan solo acceso a las operaciones de escritura a unos pocos seleccionados y los datos se registran en unos pocos servidores, pero la información se encuentra disponible para muchos usuarios. No se ejecuta en muchos nodos, solo requiere un servidor potente para procesar los datos en el backend mientras que un host frontend proporciona una interfaz. La velocidad en las bases de datos se puede optimizar a través del hardware mediante RAID Nivel 1 y mediante otras técnicas como fragmentación y reducción. En el caso de un desastre, un administrador puede revertir los cambios. El administrador maneja todo tipo de actualizaciones y seguridad, y administra todo el sistema.

- **Velocidad y volumen de transacciones:** Las bases de datos de hoy están diseñadas tanto para el procesamiento de transacciones de alto volumen como para el análisis

de datos. Esto significa que están diseñados y probados para operaciones en entornos de producción empresarial.

Ventajas de la Blockchain:

- **Descentralización:** Un sistema descentralizado es altamente tolerante a fallos. Si un nodo se bloquea en la red, no interrumpe todo el sistema. Hay otros nodos en la red que ejecutan la cadena de bloques. La descentralización también agrega más seguridad ya que la información almacenada en un nodo debe copiarse en todos los nodos de la red. Esto significa que si un nodo estuviera comprometido, un hacker necesitaría poder cambiar la información en todos los nodos de la Blockchain para manipular los datos. Esto ha demostrado ser una buena protección disuasoria frente a ataques contra el sistema.

- **Inmutabilidad:** Un Blockchain almacena información que se vuelve inmutable, lo que significa que no se puede cambiar una vez que se haya validado un bloque. Esto también lo hace resistente a la manipulación indebida y la falsificación, ya que la información se registra en un Ledger digital público almacenado en muchos nodos. Comprometer la información significa cambiar esa información en todos los nodos de la red.

- **Transparencia:** Toda la información registrada en la Blockchain es resistente a la censura. La información sobre una transacción no se puede ocultar, por lo que esto crea más confianza y agrega valor al sistema. El uso de la Blockchain no requiere permiso de nadie, es una plataforma abierta para todos en un entorno público.

- **Seguridad:** Dado que las Blockchains utilizan tecnología criptográfica avanzada y una red descentralizada distribuida, ofrecen un entorno seguro. La modificación de los datos en un bloque requiere gastar una gran cantidad de recursos informáticos. Tampoco es sencillo porque requiere cambiar los datos en todos los nodos de la red. Esto es lo que disuade a los ataques, ya que es más costoso que utilizar esos recursos para obtener recompensas por minería. Esta es una característica para ayudar a proteger la Blockchain de los mineros y hackers deshonestos.

Desventajas de las Bases de Datos:

- **Único punto de fallo:** Dado que está centralizado, hay un punto de fallo único. Los datos están en manos de una sola entidad o grupo, por lo que no hay forma de garantizar que se utilicen para el propósito correcto, como en el caso de que los datos de las redes sociales terminen en malas manos. Una empresa que tiene control de la información puede monetizarla para uso de terceros, aunque no sea lo mejor para los usuarios. Cuando se piratea una base de datos, es un grave problema, ya que puede afectar la información de muchos usuarios. Cuando un servidor de base de datos falla, también afecta a todo el sistema. Si no hay una copia de seguridad de la información almacenada en la base de datos, no hay manera de recuperar los datos. Es por esto que la conmutación por error y la redundancia son importantes en los sistemas centralizados.

- **Requiere un Administrador:** Como una base de datos requiere un administrador, si la contraseña se pierde, será más difícil recuperar una base de datos. Si el administrador de la base de datos no tiene un administrador delegado que tenga privilegios

para un sistema de administración de bases de datos, nadie puede crear nuevas bases de datos o modificar las existentes. Otro problema con esto es que cuando un administrador de base de datos abandona la empresa, se convierte en un proceso muy tedioso que tiene que restablecer las contraseñas y elevar el privilegio de un nuevo administrador. Es probable que alguien se olvide de cambiar una contraseña, eliminar ciertos privilegios o eliminar la cuenta de antiguos empleados que tienen acceso a la base de datos. Esto es algo con lo que deben lidiar los departamentos de MIS para mantener segura su información.

- **Problemas de seguridad:** En un sistema centralizado, si el administrador olvida aplicar parches y actualizaciones, el sistema puede ser vulnerable a los ataques de seguridad por parte de hackers. Esto hace que las bases de datos sean propensas a infracciones. La centralización debería simplificar la administración, pero en otras ocasiones, cuando no se realiza correctamente, puede causar problemas muy críticos que afectan la integridad de los datos en un sistema. Confiar en toda nuestra información a una sola compañía es la norma, pero puede convertirse en un problema si la empresa no cumple con las mejores prácticas de seguridad de la información. Los hackeos ya han afectado a muchas compañías importantes, y las violaciones de datos son cada vez más comunes, ya que la información es un activo valioso. Esta es la razón por la que se aplican auditorías de terceros y regulaciones estrictas para la seguridad de los datos que involucran bases de datos de producción.

Desventajas de las Blockchains:

- **Consumo de energía:** Los recursos de cómputo para ejecutar una Blockchain que utilice el algoritmo de Proof-of-Work para procesar transacciones utilizan grandes cantidades de energía. Los mineros toda la energía para resolver enigmas criptográficos para validar bloques. Cuantos más nodos tenga la minería, mayor será el esfuerzo computacional requerido para validar un bloque de transacciones.

- **Escalabilidad:** Las Blockchains no se escalan bien cuando se trata de transacciones de gran volumen. Debido al tamaño de bloque fijo, existen problemas al aumentar el volumen de transacciones. Los retrasos también afectan la velocidad de la transacción. Las soluciones de escalamiento se han convertido en el foco de muchos proyectos para optimizar el rendimiento en el manejo de más transacciones y disminuir el tiempo de procesamiento.

- **Tamaño:** Un problema con la mayoría de las bases de datos, incluidas las Blockchains, es su tamaño. Cuando crecen, consumen más espacio de almacenamiento y esto hace que disminuyan la velocidad. No es solo un problema de almacenamiento para los nodos, sino también de la red. A mayores tamaños de Blockchain se requiere más ancho de banda para transmitir a otro nodo. Esto afecta a los nuevos nodos o nodos que vuelven a estar en línea y no se han actualizado en mucho tiempo.

- **Comisiones de transacción:** Cuando la demanda es alta, las tarifas de transacción también aumentan en beneficio de los mineros. Mantener las tarifas de transacción bajas o eliminarlas es un desafío para los diseñadores de Blockchain. Con altas tarifas de transacción, los usuarios no pueden utilizar la red. Cuando los problemas de escala resuelvan los problemas con la velocidad y el volumen de las transacciones, se deben aplicar tarifas más razonables.

- **Interoperabilidad:** Esto es actualmente un problema ya que, a diferencia de las bases de datos tradicionales, cada Blockchain es en gran medida su propio ecosistema. Existen protocolos que tienen como objetivo hacer que las Blockchains interoperen entre sí. Los desarrolladores están investigando maneras de hacer que las Blockchains diferentes sean interoperables para hacer que la transferencia de valor sea mucho más simple.

Una base de datos es ideal para:

- Datos que necesitan actualización continua, como monitorización y sensores.
- Procesamiento rápido de transacciones en línea.
- Información confidencial (no transparente al público).
- Datos financieros de mercados que requieren un procesamiento rápido.
- Datos que no requieren verificación.
- Aplicaciones independientes que almacenan datos.
- Datos relacionales

Un Blockchain es ideal para:

- Transacciones monetarias
- Transferencia de valor
- Verificación de datos confiables (identidad, reputación, credibilidad, integridad, etc.)
- Verificación de clave pública
- Aplicaciones descentralizadas (DApps)
- Sistemas de votación

Base de Datos	Blockchain
Centralizado	Descentralizado
Necesita permisos de acceso	Sin necesidad de permisos
Necesita un Administrador	No necesita Administrador

Tabla 3.1 Principales diferencias entre BD y Blockchain (Tabora, 2018).

3.2 Problemas computacionales que soluciona una Blockchain

3.2.1 El problema de los generales bizantinos: El problema clásico de la computación distribuida

El problema de los **generales bizantinos** (Pease, 1995) es un experimento mental creado para ilustrar el dilema de lograr un consenso entre un conjunto de entidades con un **objetivo común** cuando entre ellas pueden existir **traidores**, es decir, entidades con objetivos

opuestos que intenten dinamitar el proceso. Ademas, se supone que las comunicaciones entre dichas entidades son limitadas e inseguras.

El problema se presenta como una analogía con un escenario de guerra, donde un grupo de generales bizantinos se encuentran acampados con sus tropas alrededor de una ciudad enemiga que desean atacar. Después de observar el comportamiento del enemigo, los generales deben **comunicar** sus observaciones y **ponerse de acuerdo** en un plan de batalla común que permita atacar la ciudad y vencer.

Para ello, los generales se comunican unicamente a través de mensajeros, existiendo la posibilidad de que algunos de los generales sean traidores y, por lo tanto, decidan enviar mensajes con **información errónea** con el objetivo de confundir a los generales leales.

Un algoritmo que solucione el problema debe asegurar que todos los generales leales acuerdan un mismo plan de acción y que unos pocos traidores no pueden conseguir que el plan adoptado por los generales leales sea equivocado (Pérez Solà y Herrera Joancomartí, 2014).

Existen ciertos riesgos que podrían impedir la victoria (Azis, 2016):

- No todos los generales son dignos de confianza; algunos pueden ser traidores
- No todos los mensajeros son dignos de confianza; algunos pueden ser traidores
- Un mensajero podría ser capturado por el enemigo y se sustituye por un mensajero traidor para transmitir noticias falsas

Una Blockchain resuelve este problema a través del protocolo de "**Consensus**", que se profundiza en el apartado 3.3, la decisión final está basada en la opinión de la **mayoría**, la solución pasa por asegurar que los planes de los traidores no afecten al consenso de los leales, con envío de planes basado en **doble vuelta**, primera vuelta los generales envían su decisión, en la segunda vuelta los generales envían los planes recibidos de otros generales (Ezpeleta y P. Álvarez, 2018).

Por tanto el algoritmo de los generales bizantinos evita inconsistencia en la información en:

- Cualquier escenario que experimente fallos por caídas, un mensajero no llega a entregar el plan de su general a los otros generales.
- Aquellos escenarios que experimenten fallos bizantinos si el número de generales es mayor o igual a **3t+1**, siendo **t** el número de generales traidores.

En la tecnología Blockchain cada general es un **nodo** con un Ledger, una copia de la Blockchain, cuando un nodo añade una transacción nueva al Ledger, el siguiente paso es **enviar** esa información a todos los demás nodos, estos nodos actualizarán su Ledger cuando reciben esta información y enviarán una "copia" de su Ledger al resto de nodos y recibirán una "copia" del resto de nodos, por tanto, si un nodo recibe una información falsa de uno de los nodos, por **comparación** con el resto de Ledgers podrá diferenciar la información falsa de la verdadera y simplemente ignorarla.

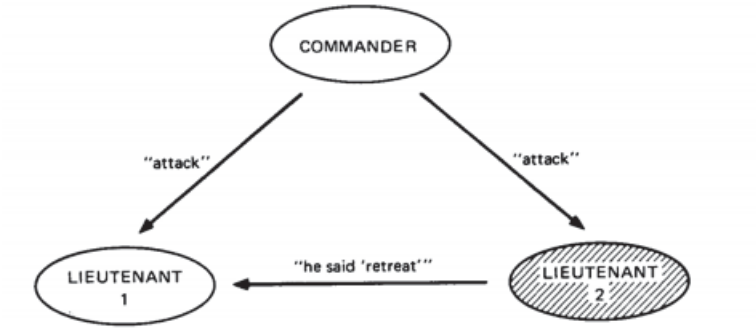


Fig. 1. Lieutenant 2 a traitor.

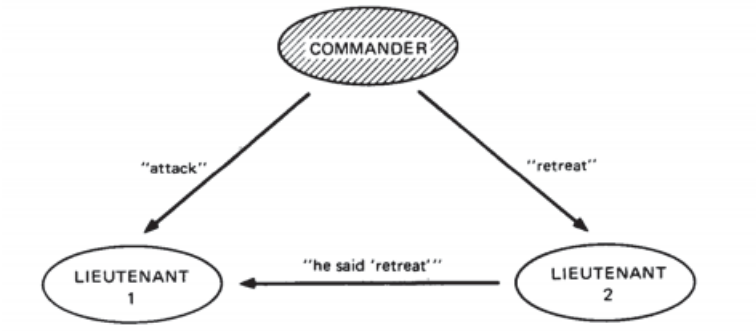


Fig. 2. The commander a traitor.

Figura 3.2 Problema Generales Bizantinos (Pease, 1995).

3.2.2 Problema del doble gasto

El **doble gasto** es una debilidad del efectivo digital, ya que permitiría gastar una moneda digital dos veces.

Dado que, a diferencia del dinero físico como las monedas, los archivos electrónicos pueden ser **duplicado** o **falsificado**, y por lo tanto el hecho de gastar una moneda digital no **elimina** sus datos de la propiedad del titular original, se necesitan otros medios para evitar el doble gasto (RSK educate, 2015).

Al igual que con el dinero falsificado, el doble gasto conlleva **inflación** dado que se crean nuevas monedas fraudulentas que anteriormente no existían. Esto devalúa la moneda en relación a otras unidades monetarias, y disminuye la **confianza** de los usuarios, así como dificulta la circulación y posesión de la moneda.

En el caso concreto de las **Blockchains**, estas se protegen contra los ataques de doble gasto **agregando** cada transacción a la Blockchain, haciendo que sea muy difícil modificarla y **verificándola** después. Para ello, utiliza un sistema **descentralizado**, basado en una gran red de nodos que **confirman** la transacción.

Es más vulnerable a este tipo de ataques, durante el inicio de la transacción en la red y por eso, a mayor número de confirmaciones, menos probable será el riesgo de sufrir un ataque (En Wikipedia, s.f.[l]).

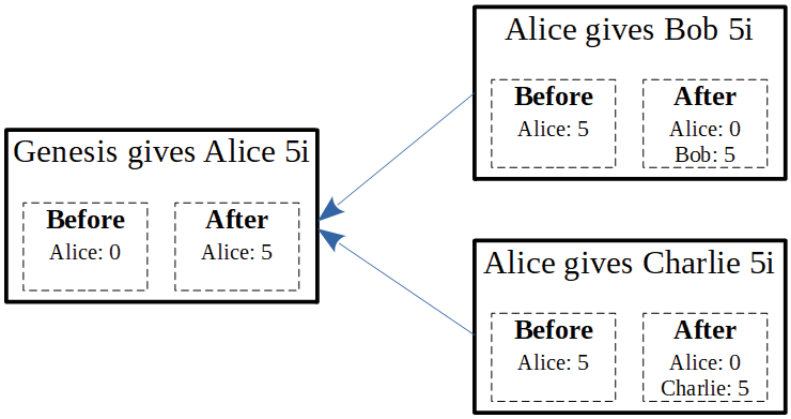


Figura 3.3 Problema Doble Gasto(Alon Gal, 2018).

Tipos de ataques:

- **Ataque de carrera**

  Esta variación del ataque de doble gasto solo se puede producir en caso de transacciones cuyos vendedores aceptan pagos **sin confirmación** ya que se exponen a que la transacción pueda ser **revertida**.

  El ataque se basa en que a la hora de realizar una transacción podemos realizar al mismo tiempo otra u otras transacciones usando los mismos fondos. El nombre del ataque viene porque se produce una carrera para ver que transacción se acepta primero.

- **Ataque Finney**

  Debe su nombre a Hal Finney que fue el primer receptor de una transacción por medio de una red Bitcoin. Solo funciona si el comerciante acepta transacciones **no confirmadas**. Requiere la participación de un minero.

  El atacante irá minando bloques, en el bloque que esté minando en ese momento, incluye una transacción que le devuelve algunas de sus monedas, pero **sin transmitir** la transacción al resto de nodos. En cada bloque que genera incluye una transferencia de una dirección A a B ambos controladas por él. Cuando encuentra un bloque hace una compra desde A y cuando se acepte el pago el atacante **difunde** su bloque, la transacción le envía las criptomonedas a él y **anulará/revertirá** el pago anterior al comerciante.

- **Ataque por fuerza bruta**

  Este ataque se puede realizar incluso siendo necesario recibir confirmaciones al realizar la transacción. El éxito o fracaso del ataque depende del Hashrate (poder de cómputo) que tenga el atacante, pero requiere un **gasto** considerable en electricidad y hardware.

  El ataque consiste en hacer una transacción en la que se **paga** al vendedor y al mismo tiempo se **mina** un Blockchain **alternativo** en el que se incluye un ataque de doble gasto, una vez se ha esperado un numero de **confirmaciones** y el vendedor **vende/envía** el producto, si en ese momento se han encontrado más de n bloques entonces se **libera** la cadena alternativa y el atacante recupera su dinero.

- **Ataque por mayoría o ataque >50 %**

  Este ataque solo se puede realizar si el atacante controla **más del 50 %** del Hashrate, este sería un caso especial del **ataque por fuerza bruta** que tendría un 100 % de probabilidades de éxito ya que el atacante va a poder generar bloques **más rápido que el resto** de la red, como se observa en la figura 3.4.

  El número de confirmaciones sería **irrelevante** pero un mayor número de confirmaciones puede **retrasar** lo suficientemente al atacante para que este método no sea rentable.

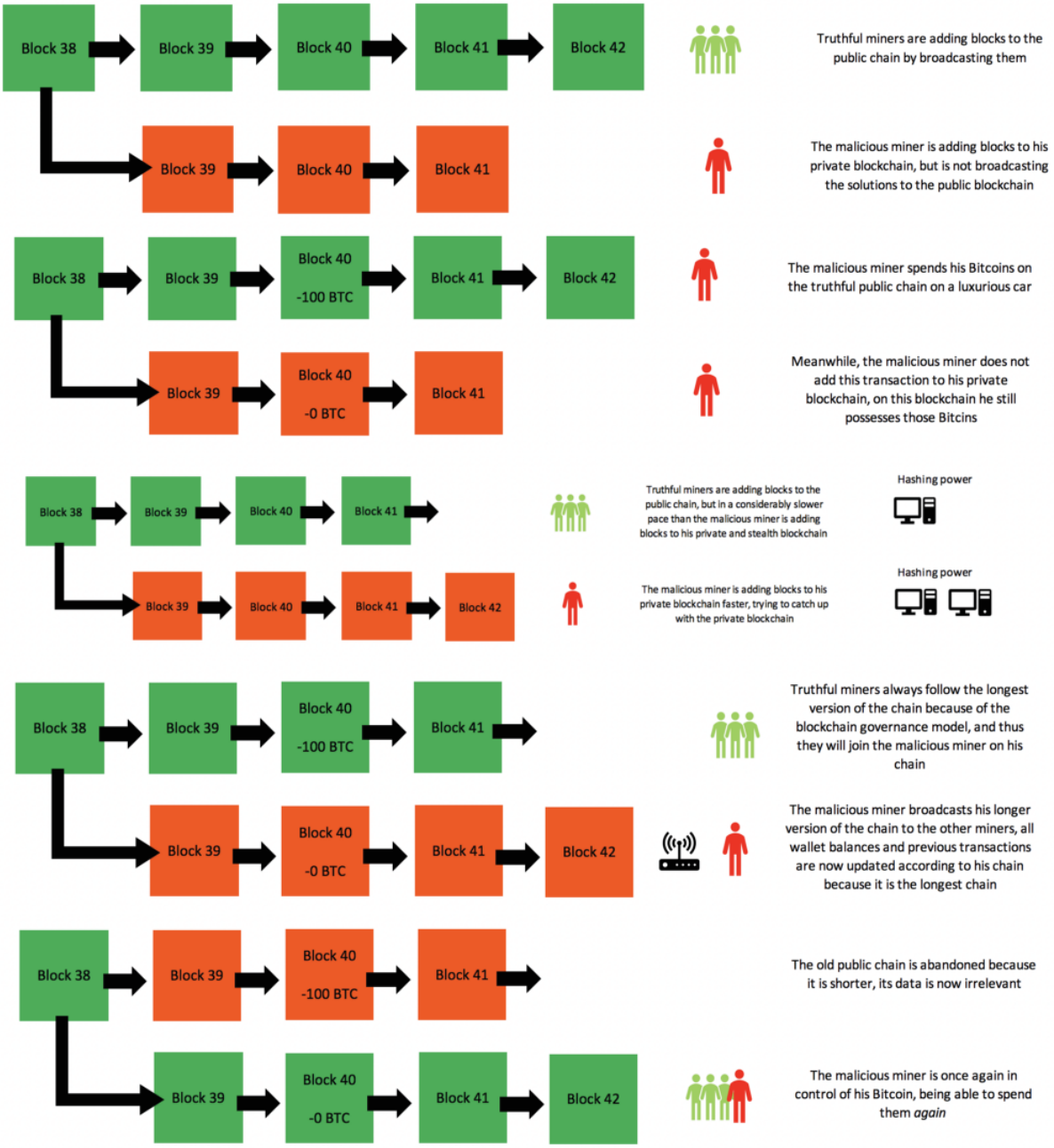


Figura 3.4 Ataque del 51 %(Jimi S., 2018).

3.3 Consensus

3.3.1 ¿Qué es el Consensus?

Consenso definición: Acuerdo producido por **consentimiento** entre todos los miembros de un grupo o entre varios grupos (En DRAE, 2018a).

Un blockchain es un sistema **descentralizado** de igual a igual (p2p) sin una figura de autoridad central. Esto crea un sistema sin **corrupción** de un único origen/central, pero todavía crea un problema importante porque una cadena de bloques no tiene "líder". Para que las cadenas de bloques tomen decisiones, deben llegar a un consenso utilizando **"mecanismos de consenso"** (Rosic, 2018).

En los últimos 30 años, los mecanismos de consenso en el mundo informático han pasado de ser una idea abstracta a la clave principal de la tecnología de bases de datos distribuida.

En los ledgers distribuidos, un mecanismo de consenso es la forma en que una mayoría o todos los miembros de la red **acuerdan** el valor de un dato o una propuesta de transacción, que luego **actualiza** el ledger.

En otras palabras, un mecanismo de consenso es un conjunto de **reglas** y procedimientos que mantienen un conjunto coherente de hechos entre los nodos participantes.

Los algoritmos de consenso permiten que las máquinas conectadas trabajen juntas como un **grupo** que incluso sobrevive si alguno de sus miembros falla. Esta tolerancia al fallo es otra gran ventaja de las Blockchains y las bases de datos distribuidos, que tienen **redundancia** incorporada.

Los protocolos de consenso o las plataformas de consenso son el **núcleo** de las tecnologías distribuidas. Existe una gran diversidad de algoritmos para generar consenso en función de **requisitos** como rendimiento, escalabilidad, consistencia, capacidad de datos, gobierno, seguridad y fallas redundancia (RSK educate, 2015; Andreas M. Antonopoulos, aantonop, 2016).

Consenso es que los nodos de la red están de acuerdo con el mismo estado de una cadena de bloques, en un sentido que lo convierte en un ecosistema de auto-auditoría(Lisk Academy, 2019).

Este es un aspecto absolutamente crucial de la tecnología, que lleva a cabo dos funciones clave:

1. Los protocolos de consenso permiten que se actualice una cadena de bloques, al tiempo que se garantiza que todos los bloques de la cadena sean verdaderos, además de mantener a los participantes incentivados.

2. Evita que cualquier entidad individual controle o descarrile todo el sistema Blockchain. El objetivo de las reglas de consenso es garantizar que se utilice y siga una cadena única.

Cuáles son los objetivos de un mecanismo de consenso Rosic (2018):

- **Búsqueda de acuerdo:** un mecanismo de consenso debe lograr el mayor acuerdo posible del grupo.

- **Colaborativo:** todos los participantes deben de trabajar juntos para lograr un resultado que ponga el interés del grupo primero.
- **Cooperativa:** todos los participantes deben trabajar en equipo en lugar de en forma individualizada y no deben poner sus propios intereses primero.
- **Igualitario:** un grupo que trata de lograr un consenso debe ser lo más igualitario posible. Lo que esto significa básicamente es que todos y cada uno de los votos tienen el mismo peso. El voto de un individuo no puede ser más importante que el de otro.
- **Inclusivo:** en el proceso de consenso debe participar el mayor número posible de individuos. No debería ser como la votación normal donde no todas las personas sienten ganas de votar porque creen que su voto no tendrá ningún peso a largo plazo.
- **Participativo:** El mecanismo de consenso debe ser tal que todos deben participar activamente en el proceso general.

3.3.2 Mecanismos de consenso disponibles y en desarrollo

Los protocolos de consenso son uno de los aspectos más importantes y revolucionarios de la tecnología Blockchain.

Estos protocolos crean un sistema de acuerdo irrefutable entre varios dispositivos a través de una red distribuida, mientras se evita la explotación indebida del sistema (RSK educate, 2015; Lisk Academy, 2019; Azis, 2016).

Proof-of-Work (PoW), prueba de trabajo

Los mineros **compiten** para agregar el siguiente bloque (un conjunto de transacciones) en la cadena de bloques, es una **carrera** para resolver un **rompecabezas** criptográfico extremadamente duro. El primero en resolver el rompecabezas, gana el **premio**. Como recompensa por sus esfuerzos, el minero recibe tokens nuevos y una pequeña comisión de transacción pagada por los usuarios.

En Blockchains grandes como Bitcoin o Etherum se consume una **cantidad importante de energía** por todos los mineros. En Bitcoin, el minero recibe 12.5 ejemplares nuevos hasta 2020 que ocurrirá el siguiente "halving", proceso que consiste en la reducción a la mitad de la recompensa de los mineros por completar un bloque de transacción, permite controlar el suministro (Bárbara Nogales, 2019), y una pequeña comisión de transacción.

El número de transacciones por segundo de este mecanismo es alrededor de **7-30** transacciones/segundo.

Soporta comisiones de transacción **altas**.

Para atacar una Blockchain (3.2.2) que use PoW se debe tener mayor potencia de cálculo que el 50 % de nodos de la red juntos, lo que equivale a grandes costes y gastos de energía para el atacante, **desincentivando** el ataque a las Blockchains que usan este protocolo de consenso, siendo uno de los protocolos **más seguros** cuando la Blockchain es grande y está suficientemente **distribuida** en multitud de nodos.

- **Directed Acyclic Graphs (DAG), Grafos Acíclicos Dirigido**

En un grafo (o red de nodos conectados) la información se puede pasar de un nodo a otro a lo largo de diferentes conexiones. La información puede dejar un nodo, pasar a través

de otros nodos y regresar al nodo original. Cuando esto ocurre sin que se haya encontrado ningún nodo más de una vez, el gráfico se puede definir como cíclico.

Los grafos acíclicos son grafos que no tienen ningún ciclo, no hay una ruta para que la información regrese a los nodos primarios de cada diagrama sin duplicar o pasar por un nodo que ya se ha encontrado.

Si se le da una dirección/sentido a las conexiones para que funcionen como una calle de **una sola vía** para la información, y se asegura que ningún nodo se conecte a ninguno de los otros nodo anteriores, nuestro grafo es ahora un **grafo acíclico dirigido**, ver figura 3.5.

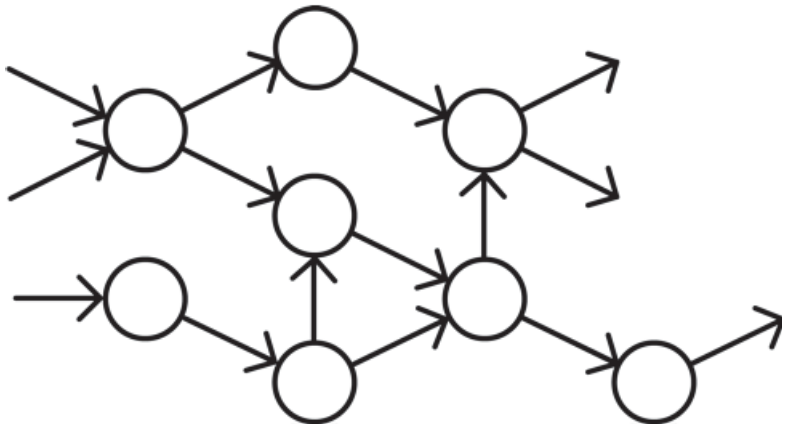


Figura 3.5 Directed Acyclic Graphs (Fantom Foundation, 2018).

Las **Blockchain** funcionan de manera que cada bloque nuevo incluye una **referencia** a la anterior. Sin embargo, se pueden crear cadenas **ramificadas** cuando dos mineros producen un bloque al mismo tiempo, o se produce un **ataque del 51 %**.

Para garantizar que la cadena principal siga siendo viable, las cadenas de bloques imponen un **período** de tiempo arbitrario entre los bloques para que la red tenga tiempo para **consolidar** y **verificar** qué "ramificación" de la cadena es correcta. Este período de espera se conoce como tiempo de bloqueo, y las ramas más cortas o malintencionadas quedan "**huérfanas**" (descontinuadas).

Los **DAG**, por otro lado, permiten que coexistan e interconecten **múltiples** cadenas de bloques. Los nodos pueden existir en **paralelo**, siempre que la información se dirija en la **misma dirección**.

Esto abre un nuevo rango de opciones de confirmación que **eliminan** la necesidad de tiempos de bloqueo, además de **reducir** la cantidad de **procesamiento/energía** desperdiciado en "cadenas huérfanas" abandonadas.

Las transacciones individuales se validan entre sí. Los usuarios de la red son mineros y validadores, aunque no pueden validar sus propias transacciones. Esto generalmente significa que en un DAG hay **poca** o **ninguna** necesidad de pagar comisiones.

Al **escalar** de manera muy eficiente y evitar o reducir las tarifas de los usuarios, los DAG son adecuados para **grandes volúmenes** de transacciones, incluidas las micro y nano-transacciones. Cuanto mayor sea el volumen de transacciones, más rápido las validará un DAG.

Los DAG también **eliminan** la necesidad de mineros y, a su vez, equipos de minería, lo que significa un **menor consumo** de energía.

El resultado final: un potencial sin precedentes para un **flujo** de información rápido y altamente escalable en una red completamente **descentralizada**. Los DAG podrán competir con las redes existentes en un futuro próximo en términos de **rendimiento** y **seguridad** (Fantom Foundation, 2018; Max Thake, 2018).

Proof-of-Stake (PoS), prueba de participación/posesión

En este tipo de algoritmo de consenso, en lugar de invertir en equipos informáticos potentes y caros para participar en una carrera para minar bloques, un **'validador'** invierte en los **tokens** del sistema.

Es necesario tener en cuenta el término validador, el uso de este término es debido a que no existe **creación** de moneda (minería) en **PoS**. En su lugar, todas las monedas existen desde el primer día y los validadores, que tienen su **participación "bloqueada"** en el sistema, solo reciben las **comisiones** de transacción como **pago**.

Como prueba de la participación, su **probabilidad** de ser elegido para crear el siguiente bloque y recibir el pago, depende de la **fracción** de monedas en el sistema que **posees**. Un validador con 300 monedas tiene tres veces más probabilidades de ser elegido que alguien con 100 monedas.

Una vez que un validador crea un bloque, ese bloque aún debe **añadirse** al Blockchain. Los diferentes sistemas de PoS varían en la forma en que regulan este paso.

En Tendermint, por ejemplo, cada nodo en el sistema tiene que firmar en un bloque hasta que se alcance un voto mayoritario, mientras que en otros sistemas, se elige un grupo aleatorio de firmantes.

Este mecanismo de consenso tiene un consumo de energía **bajo**, ya que no requiere ningún hardware potente para realizar cálculos.

El número de transacciones por segundo de este mecanismo es alrededor de **30-170** transacciones/segundo.

Comisiones de transacción **bajas**.

Para atacar una Blockchain (3.2.2) que use PoS se debe poseer una gran cantidad de tokens/monedas de esta Blockchain para tener más posibilidades de ser el validador, eso equivale a un desembolso inicial grande para adquirir esos tokens, y tras realizar el ataque la Blockchain pierde **confianza** y **valor** lo que haría bajar mucho el valor de la inversión inicial, sería similar a robar en tu empresa siendo dueño de ella.

Delegated proof of stake (DPoS), prueba de participación/posesión delegada

Intenta combinar las características de la prueba de participación y la prueba de trabajo. Utiliza un proceso de **votación descentralizada** a través de lo que se conoce como "**testigos**" como una forma de mitigar la centralización de la red en el caso del PoS.

Tiene la **flexibilidad** de los parámetros de la cadena de bloques, por ejemplo, tarifas, número de testigos, intervalo de bloque, recompensas de bloque, etc.

Son configurables por el "**comité**", que es un grupo de partes interesadas/nodos **elegidos** de entre los testigos, que no reciben **ninguna** recompensa, pero con la **capacidad** para manipular los parámetros globales de la cadena de bloques mediante **votación** aplicarlos posteriormente en un mantenimiento de la Blockchain.

Este mecanismo de consenso tiene un consumo de energía **muy bajo**, menor que el PoS.

El número de transacciones por segundo de este mecanismo es alrededor de **25-2500** transacciones/segundo, a cambio tiene una estructura más **centralizada**.

Comisiones de transacción **bajas**.

Leader-Based Consensus (LBC), consenso basado en el líder

Los nodos eligen temporalmente un nodo para ser un **líder**. El líder es entonces responsable de **validar** transacciones.

Round Robin (RR), Robin circular

Se usa en Blockchains **privadas** donde la minería está **restringida** a un conjunto de entidades **identificables**.

Resuelve el dilema en las que un participante puede monopolizar el proceso minero. La solución se encuentra en una restricción en el número de bloques que puede ser creado por el mismo minero dentro de una ventana de tiempo dada. Esto hace que se cumpla un "horario Round Robin", en el que los mineros autorizados deben crear bloques en **orden de rotación** para generar un blockchain válido.

Federated Consensus (FC) / Federated Byzantine Agreement (FBA), Consenso Federado/ Acuerdo Bizantino Federado

El Consenso Federado logra solidez a través de "**segmentos de quórum**" (quorum slice), decisiones de **confianza individual** hecho por cada nodo que **juntos** determinan el quórum a nivel del sistema. Los segmentos enlazan el sistema de la misma forma en que las decisiones y el tránsito de las redes individuales unifican Internet.

FC tiene **requisitos** informáticos y financieros **modestos** en comparación con los PoW y PoS.

Hay varias diferencias importantes entre el acuerdo bizantino no federado tradicional y el acuerdo bizantino federado (FBA). El acuerdo bizantino garantiza un **consenso distribuido** a pesar del fallo bizantino de los participantes. Sin embargo, requiere un acuerdo unánime sobre la membresía del sistema por parte de todos los participantes. Cada nodo en la red debe ser **conocido y verificado** con anticipación.

El acuerdo bizantino federado (FBA) ofrece membresía **abierta** y control descentralizado al acuerdo bizantino.

La diferencia clave entre un sistema de acuerdo bizantino y un sistema de acuerdo bizantino federado (FBA) es que en FBA cada nodo elige sus propios segmentos de quórum. Los quórums de todo el sistema resultan de estas decisiones por nodos individuales. En FBA, no hay un **controlador** de acceso ni una **autoridad centralizada**, por lo que los nodos individuales deciden en qué otros participantes confían para obtener información.

Los nodos pueden tener múltiples segmentos, y estas elecciones de nodo individuales pueden basarse en criterios **extrínsecos**. Por ejemplo, un banco en particular puede verse como de buena reputación, lo que hace que otros nodos requieran su validación de todas las transacciones; es posible que una empresa ya tenga una relación financiera con una cooperativa de crédito y quiera asegurarse de que tanto el banco como la entidad firmen todas las transacciones (Stellar Development Foundation, 2015).

Este mecanismo de consenso tiene un consumo de energía **muy bajo**.

El número de transacciones por segundo de este mecanismo es alrededor de **100-2500** transacciones/segundo, tiene una estructura **descentralizada**.

Comisiones de transacción **muy bajas**.

Practical Byzantine Fault Tolerance (PBFT), tolerancia práctica al fallo bizantino

Cada nodo mantiene un **estado interno** (información específica o estado en curso). Cuando un nodo recibe un mensaje, lo utiliza junto con su estado interno para ejecutar un cálculo u operación. Este cálculo, a su vez, le dice a ese nodo individual qué pensar sobre el mensaje en cuestión. Luego, después de llegar a su **decisión individual** sobre el nuevo mensaje, ese nodo **comparte** esa decisión con todos los demás nodos del sistema. La **decisión** de consenso se determina con base en las **decisiones totales** enviadas por todos los nodos.

Entre otras consideraciones, este método de establecer consenso **requiere menos esfuerzo** que otros métodos a costa de **perder el anonimato** en el sistema.

Hay otras variaciones de la PBFT:

- **PBFT Derivada** (Hyperledger)
- **RBFT** - Tolerancia a fallos bizantinos redundantes - (Evernym)
- **SBFT** - Tolerancia a fallos bizantinos simplificada - (Cadena)

Proof-of-Activity, prueba de actividad

Es un enfoque **híbrido** que combina PoW y PoS.

La minería se inicia como la manera tradicional de **PoW**, con los mineros compitiendo para **resolver** un enigma criptográfico. Los **bloques** extraídos no contienen ninguna transacción (son como **plantillas**), por lo que el bloque ganador solo contendrá un encabezado y la dirección de recompensa del minero. En este punto, el sistema cambia a **PoS**. Basado en la información del encabezado, se elige un grupo aleatorio de **validadores** para **firmar** el nuevo bloque. Cuantos más tokens tenga el validador en el sistema, más probable es que sea elegido. La **plantilla** se convierte en un **bloque** de pleno derecho tan pronto como todos los validadores lo **firmen**. Si algunos de los validadores seleccionados no están disponibles para completar el bloque, entonces se selecciona el siguiente bloque ganador, se elige un nuevo grupo de validadores, y así sucesivamente, hasta que un bloque reciba la cantidad correcta de firmas.

Las **comisiones** se dividen entre el **minero** y los **validadores** que firmaron el bloque.

Las críticas de la prueba de actividad son las mismas que para la prueba de trabajo (se requiere demasiada **energía** para extraer bloques) y prueba de participación (no hay nada que impida que un validador firme doble).

Proof-of-Burn, prueba de quema/eliminación

Con la prueba de quema, en lugar de gastar dinero en costosos equipos informáticos o energía, se **'queman'** tokens enviándolos a una dirección donde sean **irrecuperables**. Al comprometer las monedas a la nada, el nodo gana un **privilegio** de por vida para **minar** en el sistema basado en un proceso de selección al azar.

Dependiendo de cómo se implementa la prueba de quema, los mineros pueden quemar la moneda nativa o la moneda de una cadena alternativa, como bitcoin. Cuantas **más monedas** quemas, **más posibilidades** tienes de ser seleccionado para minar el siguiente bloque.

Con el tiempo, su participación en el sistema decae, por lo que eventualmente querrá quemar más monedas para **aumentar** las posibilidades de ser seleccionado en el proceso. (Esto imita el proceso minero de bitcoin, donde es necesario invertir continuamente en equipos de computación más modernos para mantener el poder de hashing).

Si bien la prueba de quema es una alternativa interesante a la prueba de trabajo, el protocolo **desperdicia** recursos innecesarios. Otra crítica es que el poder minero simplemente va a aquellos que están dispuesto a quemar más dinero.

La única Blockchain que utiliza la prueba de quemadura es slimcoin, una criptomoneda basada en peercoin. Utiliza un combinación de prueba de trabajo, prueba de participación y prueba de quema.

Proof-of-Capacity, prueba de capacidad/espacio

En este protocolo se 'paga' con **espacio** en el disco duro. Cuanto **más espacio** de disco duro tenga, **mayor** será su **probabilidad** de minar el siguiente bloque y ganar la **recompensa** del bloque.

Antes de minar en un sistema de prueba de capacidad, el algoritmo genera grandes conjuntos de datos conocidos como 'parcelas/plots', que almacena en el disco duro del minero. Cuantos más plots tenga, más posibilidades tendrá de encontrar el siguiente bloque.

Al invertir en terabytes de espacio en el disco duro, se **'compra'** una mayor **posibilidad** de crear bloques duplicados y bifurcar el sistema.

Con la prueba de capacidad, existe el problema de no haber nada en juego para **disuadir** a los actores maliciosos.

Las variantes de la prueba de capacidad incluyen la prueba de almacenamiento y la prueba de espacio.

Burstcoin es el único Blockchain que utiliza una forma de prueba de capacidad.

Proof-of-Elapsed time, prueba de tiempo transcurrido

El fabricante de chips Intel ha desarrollado su propio protocolo de consenso alternativo llamado **prueba de tiempo transcurrido**.

Este sistema funciona de manera similar a la prueba de trabajo, pero consume mucha **menos** electricidad. Además, en lugar de que los participantes resuelvan un enigma criptográfico, el algoritmo utiliza un entorno de ejecución de confianza (TEE), como SGX, para **garantizar** que los bloques se produzcan de forma **aleatoria** pero **sin el trabajo** requerido.

El enfoque de Intel se basa en un tiempo de espera garantizado proporcionado a través del TEE. Según Intel, el algoritmo es escalable a miles de nodos y se ejecutará de manera eficiente en cualquier procesador Intel que soporte SGX.

El único problema con este protocolo es que requiere que se **confíe** en **Intel** y el objetivo es evitar tener que confiar en terceros de lo que intentábamos escapar con las blockchains públicas.

Resumen de los distintos mecanismos de consenso

Nombre	Abr.	Concepto	Blockchain
Proof-of-Work	PoW	Protocolo criptográfico descentralizado p2p. Sin autoridad central. Nodos "honestos" controlan la mayoría del poder computacional. Son públicos o sistemas sin permiso: No es necesario saber quién es el otro. Minado exitoso solo posible realizando el trabajo computacional.	Bitcoin, Ethereum, Colored Coins, Proprietary Metacoins, Factom, Coinprism, Litecoin, IOTA (DAG)
Proof-of-Stake	PoS	La validación requiere que el nodo tenga tokens. El nodo puede minar de acuerdo con cuantos tokens ya tenga.	Ethereum (Casper - 2019), Peercoin, Blackcoin and NXT
Delegated proof of stake	DPoS	Intenta combinar la PoS y características de la PoW. Utiliza un proceso de votación descentralizado a través de los testigos como una forma de mitigar el potencial centralización de la red. Tiene flexibilidad de la blockchain. Parámetros (configurables por el comité)	Graphene, Steem, BitShares
Leader-based consensus	LBC	Los nodos eligen temporalmente un nodo para ser un líder. El líder es responsable de validar las transacciones.	BigChainDB, Tangaroa, Juno (JPMC), Juno permite cifrado en cualquier método que el usuario prefiera.
Round Robin	RR	Blockchain privado. Un conjunto de mineros rota en orden para minar los bloques en un determinado período de tiempo.	MultiChain, Tendermint

Nombre	Abr.	Concepto	Blockchain
Federated Consensus / Federated Byzantine Agreement	FC / FBA	Robustez a través de la confianza individual. Decisiones tomadas por cada nodo que juntos determinan el nivel de quórum del sistema	Ripple, Stellar
Practical Byzantine Fault Tolerance	PBFT	Proceso computacional interno con el estado del nodo y el mensaje. Se comparan los resultados de cada uno de los nodos y se obtiene un consenso. En base a los resultados de todos los nodos.	Fabric (Hyperledger - Derived PBFT), Evernym (Redundant BFT), Chain (Simplified BFT)
Proof-of-Actvity	PoA	La prueba de actividad es un híbrido. enfoque que combina ambas PoW y PoS.	Decred
Proof-of-Burn	PoB	Al entregar tus monedas al sistema, obtienes privilegio de minar en el sistema basado en un proceso de selección aleatoria.	Slimcoin
Proof-of-Capacity	PoC	Cuanto más espacio de disco duro disponible, mayor probabilidad de minado del siguiente bloque y ganar la recompensa del bloque.	Burstcoin
Proof-of-Elapsed Time (by Intel)	PoET	Es un protocolo de elección al azar. Se basa en entornos de ejecución de confianza proporcionado por el SGX de Intel. Abordar la necesidades de grandes poblaciones de Participantes.	Sawtooth Lake Project (Hyperledger)

Tabla 3.6 Comparativa de Mecanismos de Consenso (RSK educate, 2015).

3.4 criptografía

Criptografía definición: Arte de escribir con clave **secreta** o de un modo enigmático (En DRAE, 2018b).

La criptografía es el ámbito de la criptología que se ocupa de las técnicas de **cifrado**

o codificado destinadas a alterar las representaciones lingüísticas de ciertos mensajes con el fin de hacerlos ininteligibles a receptores **no autorizados** (En Wikipedia, s.f.[g]).

Objetivos:

- Confidencialidad
- Integridad
- Vinculación
- Autenticación

Las principales técnicas criptográficas que se usan en la tecnología Blockchain son (CS-Breakdown, 2015b):

- Criptografía asimétrica - par de llaves privada/pública
- Criptografía de curvas elípticas
- Funciones hash - SHA-256

3.4.1 Criptografía simétrica

La criptografía **simétrica**, también llamada criptografía de **clave secreta** o criptografía de una clave, es un método criptográfico en el cual se usa una **misma** clave para **cifrar y descifrar** mensajes en el emisor y el receptor. Las dos partes que se comunican han de ponerse de acuerdo de **antemano** sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente **cifra** un mensaje usando la clave, lo **envía** al destinatario, y éste lo **descifra** con la misma clave, ver figura 3.6.

Seguridad

Un buen sistema de cifrado pone toda la **seguridad** en la **clave** y ninguna en el algoritmo.

Dado que toda la seguridad está en la clave, es importante que sea muy difícil de averiguar. El espacio de **posibilidades** de claves debe ser amplio.

Actualmente, los ordenadores pueden descifrar claves con extrema rapidez, y ésta es la razón por la cual el **tamaño** de la clave es importante en los criptosistemas modernos. Algoritmos de cifrado de diseño más reciente como 3DES, Blowfish e IDEA usan claves de **128 bits**, lo que significa que existen $\mathbf{2^{128}}$ **claves** posibles. Esto equivale a muchísimas posibles claves, y aun en el caso de que una gran cantidad de máquinas estuvieran cooperando para descifrarla por fuerza bruta, tardarían bastante **tiempo** en encontrar la clave secreta.

Inconvenientes

El principal problema con los sistemas de cifrado simétrico está ligado al **intercambio/distribución** de claves. Una vez que el remitente y el destinatario hayan intercambiado las claves pueden usarlas para comunicarse con seguridad, pero ¿qué canal de comunicación que sea seguro han usado para **transmitirse** las claves?

Sería mucho más fácil para un atacante intentar **interceptar** una clave que probar las posibles combinaciones del espacio de claves (En Wikipedia, s.f.[j]).

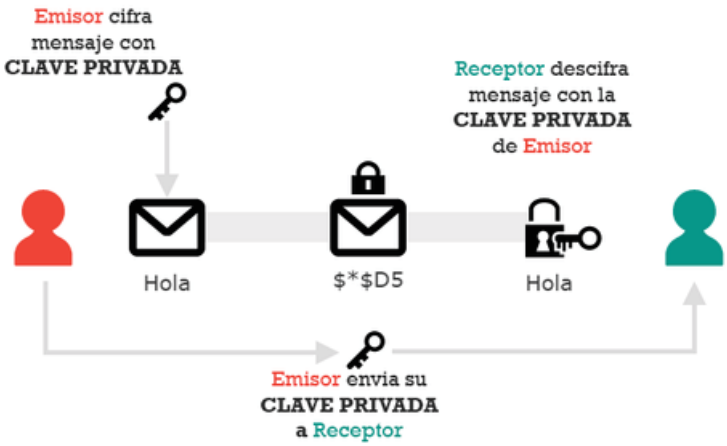


Figura 3.6 Funcionamiento criptografía Simétrica (Dolores Fuentes, 2016).

3.4.2 Criptografía asimétrica

La criptografía **asimétrica**, también llamada criptografía de clave **pública**, es el método criptográfico que usa un **par de claves/llaves** para el envío de mensajes. Las dos claves pertenecen a la misma persona.

Una clave es **pública** y se puede **entregar** a cualquier persona, la otra clave es **privada** y el propietario debe **guardarla** de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

Los 'sistemas de cifrado de clave pública' o 'sistemas de cifrado asimétricos' se inventaron con el fin de **evitar** por completo el problema del **intercambio de claves** de los sistemas de cifrado simétricos. Con las claves públicas no es necesario que el remitente y el destinatario se pongan de acuerdo en la clave a emplear. Todo lo que se requiere es que, antes de iniciar la comunicación secreta, cada uno debe **conseguir** la llave **pública** del otro y **cuidar** cada uno su llave **privada**. Es más, esas mismas claves **públicas** pueden ser usada por **cualquiera** que desee comunicarse con alguno de ellos (En Wikipedia, s.f.[h]; Xavier Decuyper, Simply Explained Savjee, 2017; Bill Buchanan OBE, 2018).

Se estudia en profundidad en el apartado 3.5.

Cifrado de clave pública

Un mensaje **cifrado** con la clave **pública** de un destinatario no puede ser **descifrado** por nadie (incluyendo al que lo cifró), excepto el poseedor de la clave **privada** correspondiente, presumiblemente su propietario y la persona asociada con la clave pública utilizada. Su función es garantizar la **confidencialidad** del mensaje.

Si una persona que emite un mensaje a un destinatario, usa la llave pública de este último para cifrarlo; una vez cifrado, sólo la clave privada del destinatario podrá descifrar el mensaje, ya que es el único que debería conocerla. Por tanto se logra la confidencialidad del envío del mensaje, nadie salvo el destinatario puede descifrarlo. Cualquiera, usando la

llave pública del destinatario, puede cifrarle mensajes; los que solo serán descifrados por el destinatario usando su clave privada.

Firma Digital

Un mensaje **firmado** con la clave **privada** del remitente puede ser **verificado** por cualquier persona que tenga acceso a la clave **pública** de dicho remitente, lo que demuestra que este **remitente** tenía acceso a la clave **privada** (y por lo tanto, es probable que sea la persona asociada con la clave pública utilizada).

Se asegura así que el mensaje no ha sido **alterado**, puesto que cualquier manipulación del mensaje repercutiría en un resultado erróneo del algoritmo de resumen del mensaje. Se utiliza para garantizar la autenticidad del mensaje.

Si el propietario del par de claves usa su clave privada para cifrar un mensaje, cualquiera puede descifrarlo utilizando la clave pública del primero. En este caso se consigue la identificación y autentificación del remitente, ya que se sabe que sólo pudo haber sido él quien empleó su clave privada (salvo que un tercero la haya obtenido).

Esta idea es el fundamento de la **firma electrónica**, donde jurídicamente existe la presunción de que el firmante es efectivamente el dueño de la clave privada.

Seguridad

Toda la seguridad debe descansar en la **clave** y no en el algoritmo. Por lo tanto, el tamaño de la clave es una medida de la seguridad del sistema, pero no se puede comparar el tamaño de la clave del cifrado simétrico con el del cifrado de clave pública para medir la seguridad.

En un ataque de fuerza bruta sobre un cifrado simétrico con una clave del tamaño de 80 bits, el atacante debe **probar** hasta 2^{80}-1 claves para encontrar la clave correcta.

En un ataque de fuerza bruta sobre un cifrado de clave pública con una clave del tamaño de 512 bits, el atacante debe **factorizar** un número compuesto codificado en 512 bits (hasta 155 dígitos decimales).

La **cantidad de trabajo** para el atacante será **diferente** dependiendo del cifrado que esté atacando. Mientras **128 bits** son suficientes para cifrados **simétricos**, dada la tecnología de factorización de hoy en día, se recomienda el uso de **claves públicas de 1024 bits** para la mayoría de los casos (En Wikipedia, s.f.[h]).

Ventajas e inconvenientes

La mayor ventaja de la criptografía asimétrica es que la **distribución** de claves es más **fácil** y **segura** ya que la clave que se distribuye es la pública manteniéndose la privada para el uso exclusivo del propietario, pero este sistema tiene desventajas:

- Para una misma longitud de clave y mensaje se necesita mayor tiempo de proceso.
- Las claves deben ser de mayor tamaño que las simétricas.
- El mensaje cifrado ocupa más espacio que el original.

Los nuevos sistemas de clave asimétrica basado en **curvas elípticas** tienen características menos costosas.

Herramientas como PGP, SSH o la capa de seguridad SSL para la jerarquía de protocolos TCP/IP utilizan un **híbrido** formado por la criptografía asimétrica para intercambiar claves de criptografía simétrica, y la criptografía simétrica para la transmisión de la información.

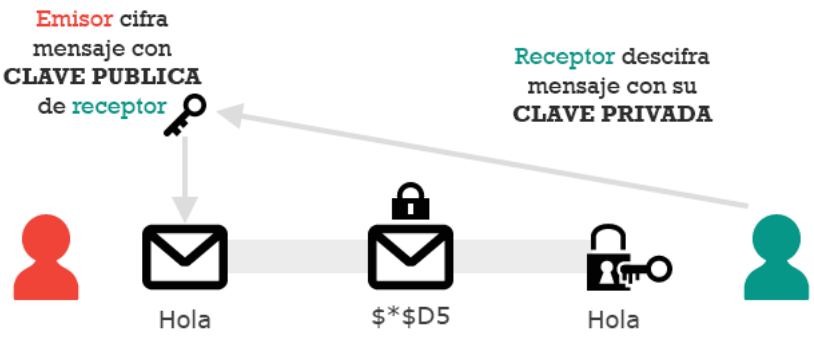


Figura 3.7 Funcionamiento criptografía Asimétrica (Ivan, 2017).

3.4.3 Criptografía de Curva Elíptica (CCE)

La **Criptografía de Curva Elíptica, CCE** (del inglés: **Elliptic curve cryptography, ECC**) es una variante de la criptografía **asimétrica** o de clave pública basada en las **matemáticas** de las curvas elípticas.

Sus autores argumentan que la CCE es más **rápida** y usa claves más **cortas** que otros métodos, como RSA, al tiempo que proporcionan un nivel de seguridad **equivalente**. La utilización de curvas elípticas en criptografía fue propuesta de forma independiente por Neal Koblitz y Victor Miller en 1985 (Koblitz, 1987; Miller, 1985).

Los sistemas de criptografía asimétrica o de clave pública utilizan dos claves distintas: una de ellas es pública, la otra es privada. La posesión de la clave pública no proporciona suficiente información para determinar cuál es la clave privada.

Este tipo de sistemas se basa en la **dificultad** de encontrar la solución a ciertos problemas matemáticos. Uno de estos problemas es el llamado **Problema del Logaritmo Discreto** (PLD).

Problema del Logaritmo discreto

> Encontrar el valor de b dada la ecuación $a^b = c$, cuando a y c son valores conocidos, puede ser un problema de complejidad **exponencial** para ciertos grupos finitos de gran tamaño; mientras el problema inverso, la exponenciación discreta puede ser evaluado eficientemente usando por ejemplo exponenciación binaria.
>
> En álgebra abstracta, se conoce como logaritmo discreto de y en base g, donde g e y son elementos de un grupo cíclico finito G, a la solución x de la ecuación $g^x = y$. Esto, se puede denotar matemáticamente como:
>
> $x = \log_g(y) \iff g^x = y$
>
> Mientras que el cálculo de su inversa — la exponenciación discreta — es una tarea muy sencilla en términos computacionales, el cálculo del **logaritmo discreto** no es sencillo en muchos grupos.
>
> El hecho de que el problema sea «**irresoluble**» en un tiempo razonable si se utiliza aritmética modular hace que esto se use en criptografía, en el método de intercambio de claves de Diffie-Hellman o en el sistema de El Gamal (En Wikipedia, s.f.[p]).

Una curva elíptica es una curva plana definida por una ecuación de la forma $E(q)$:
$y^2 = x^3 + ax + b\,(mod\,q)$ ver figura 3.8.

Con el conjunto de puntos G que forman la curva (i.e., todas las soluciones de la ecuación más un punto O, llamado punto en el infinito) más una operación aditiva, se forma un grupo abeliano. Si las coordenadas x e y se escogen desde un cuerpo finito, entonces estamos en presencia de un grupo abeliano finito.

El **problema del logaritmo discreto** sobre este conjunto de puntos (**"Problema del Logaritmo Discreto sobre Curvas Elípticas**", PLDCE) se cree que es más **difícil** de resolver que el correspondiente a los cuerpos finitos.

De esta manera, las **longitudes** de claves en criptografía de curva elíptica pueden ser más **cortas** con un nivel de **seguridad** comparable.

El principal beneficio de la criptografía de curva elíptica es un tamaño de clave más **pequeño**, lo que reduce los requisitos de almacenamiento y transmisión, es decir, que un grupo de curvas elípticas podría proporcionar el **mismo** nivel de seguridad que ofrece un sistema basado en RSA con un módulo mayor y una clave mayor: por ejemplo, para una clave pública de curva elíptica de **256 bits** proporciona una seguridad comparable a una clave pública RSA de **3072 bits**.

La seguridad de la CCE depende de la capacidad de calcular una multiplicación de puntos y de la **incapacidad** de calcular el multiplicando dados los puntos originales y de producto. El tamaño de la curva elíptica determina la dificultad del problema (CSBreakdown, 2015a; En Wikipedia, s.f.[i]; En Wikipedia, s.f.[m]).

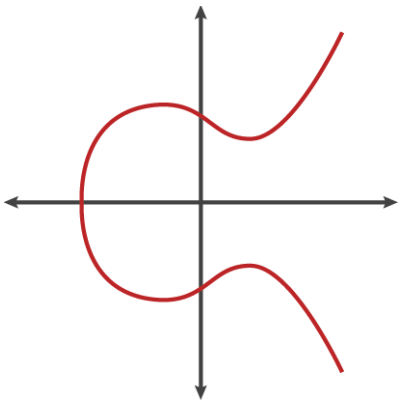


Figura 3.8 Curva Elíptica (Nick Sullivan, 2013).

Parámetros de la curva

Para utilizar **CCE**, todas las partes deben ponerse de acuerdo sobre los parámetros que definen la curva elíptica, es decir, los parámetros de dominio.

El plano está definido por p en el caso principal y el par de m y f en el caso binario.

La curva elíptica se define por las constantes a y b utilizadas en su ecuación de definición.

Finalmente, el subgrupo cíclico está definido por su generador (también conocido como punto base) G.

Para la aplicación criptográfica, el orden de G, que es el número positivo más pequeño n, tal que $nG = \mathbb{O}$ (el punto en el infinito de la curva y elemento identidad), normalmente

es primo. Dado que n es el tamaño de un subgrupo de $E(\mathbb{F}_p)$ se deduce del teorema de **Lagrange** que el número $h = \frac{1}{n}|E(\mathbb{F}_p)|$ es un entero.

En aplicaciones criptográficas, este número h, llamado cofactor, debe ser pequeño $h \leq 4$ y, preferiblemente, $h = 1$.

Para resumir: en el caso principal, los parámetros del dominio son (p,a,b,G,n,h); en el caso binario, son (m,f,a,b,G,n,h).

Varios organismos estándar publicaron parámetros de dominio de curvas elípticas para varios tamaños de planos comunes. Dichos parámetros de dominio se conocen comúnmente como "**curvas estándar**" o "**curvas con nombre**" (En Wikipedia, s.f.[m]; CSBreakdown, 2015a).

Por ejemplo los parámetros usados para generar las claves públicas en la Blockchain de Bitcoin son los correspondientes con la curva $secp256k1$ definida en *"Standards for Efficient Cryptography (SEC)"* (bitcoin.it, 2019).

Los parámetros del dominio de la curva elíptica sobre F_p asociados con una curva de Koblitz $secp256k1$, ver figura 3.9, están especificados por el sextuplo $T = (p,a,b,G,n,h)$ donde el plano finito F_p se define por:

- $p = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF$
  $FFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F$
  $= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$

La curva $E : y^2 = x^3 + ax + b$ sobre F_p se define por:

- $a = 0000000000000000000000000000000000000000000000000000000000000000$
- $b = 0000000000000000000000000000000000000000000000000000000000000007$

El punto base G en forma comprimida es:

- $G = 0279BE667EF9DCBBAC55A06295CE87$
  $0B07029BFCDB2DCE28D959F2815B16F81798$

y en forma no comprimida es:

- $G = 0479BE667EF9DCBBAC55A06295CE870B$
  $07029BFCDB2DCE28D959F2815B16F81798483$
  $ADA7726A3C4655DA4FBFC0E1108A8FD17B448$
  $A68554199C47D08FFB10D4B8$

Finalmente el orden n de G y el cofactor son:

- $n = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF$
  $EBAAEDCE6AF48A03BBFD25E8CD0364141$
- $h = 01$

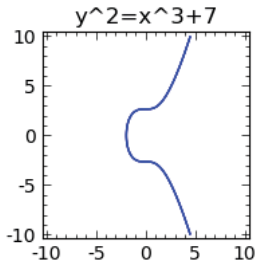


Figura 3.9 Curva Elíptica secp256k1 (Shirriff, 2014).

Propiedades de la curva

1. Simetría **horizontal**. Cualquier punto de la curva se puede reflejar sobre el eje x y permanecer en la misma curva.

2. Cualquier línea no vertical interesectará la curva en un máximo de **tres** lugares.

3. Trazando en la curva una línea con dos puntos A y B la intersectará en otro punto más C' cuyo reflejo intersectará la curva en el punto C. Cualquiera de los dos puntos en una curva se pueden unir para obtener un nuevo punto. Trazando la recta del punto C al A y reflejando el punto que intersecta la curva se obtiene el punto D, y así sucesivamente. Así es como se realiza la operación **aditiva** en curvas elípticas, de manera que $A + B = C$, observe la figura 3.10.

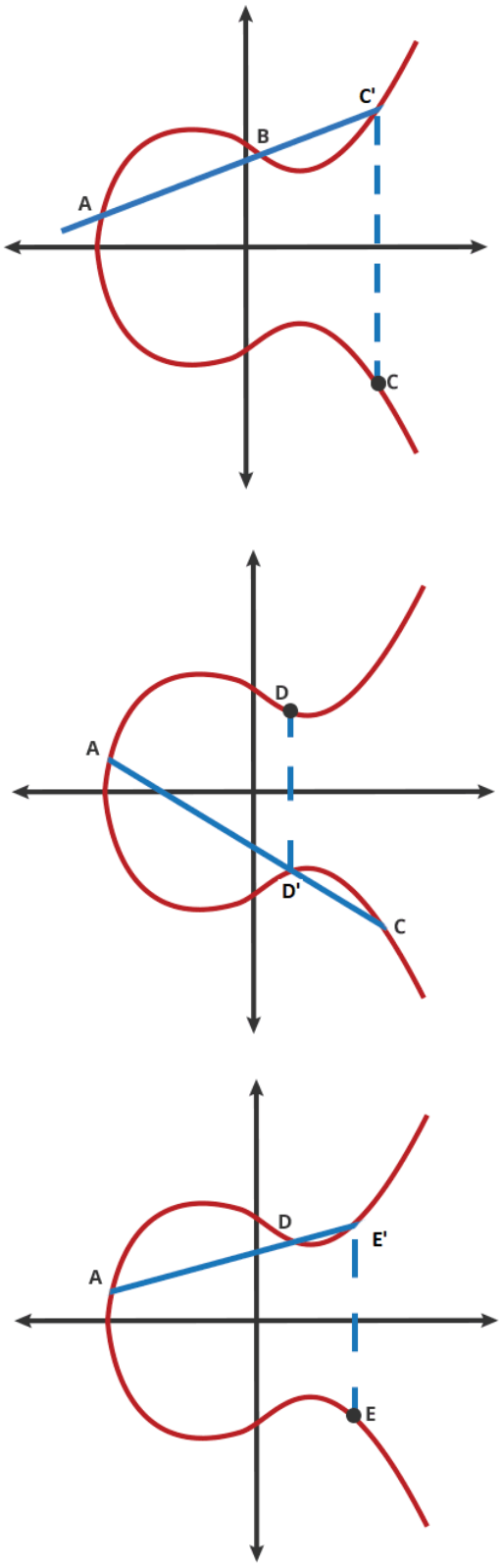


Figura 3.10 Propiedades Curva Elíptica (Nick Sullivan, 2013).

Multiplicación de CCE

Si se tiene un punto de la curva G y se **suma** consigo mismo $G+G=2G$, equivale a trazar la tangente a G, donde intersecte con la curva será el punto $-2G$ y su reflejo será $2G$, y así sucesivamente se obtiene la **multiplicación** de G como suma de k veces el punto G, ver figura 3.11.

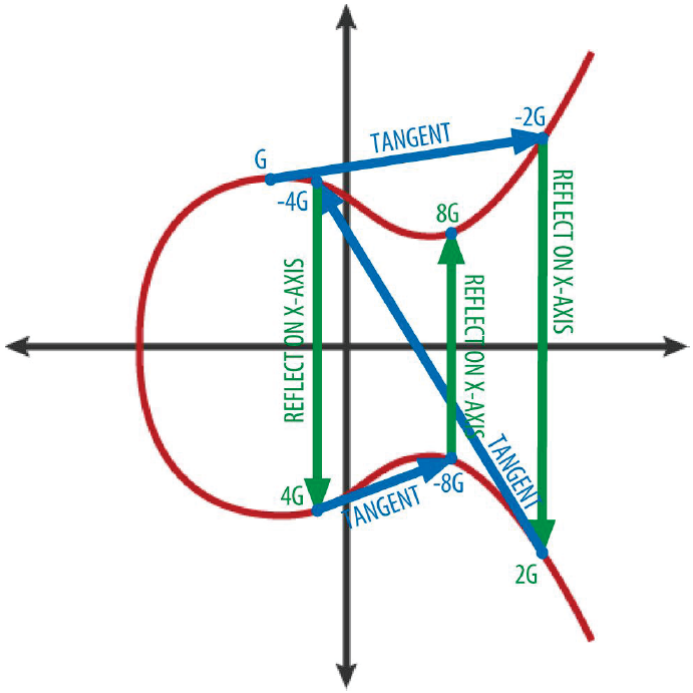


Figura 3.11 Multiplicación Curva Elíptica (RSK educate, 2015).

4. Si se tienen dos puntos, un **punto inicial** G y ese mismo punto multiplicado consigo mismo k veces para llegar a un **punto final** $P_f = kG$, encontrar k cuando solo se conoce el punto inicial y el punto final es muy difícil.
   Calcular el punto P_f conociendo k y G es una operación **sencilla**, pero conociendo solo G yP_f no se puede determinar la cantidad de saltos/multiplicaciones que ha sufrido G para llegar a P_f, es necesario realizar el cálculo de P_f multiplicando n veces el número G hasta conseguir llegar a P_f, por tanto la propiedad más interesante de las curvas elípticas es que no existe la operación de división, de manera que $k \neq P_f/G$, **fácil de hacer, difícil de deshacer** (Nick Sullivan, 2013; RSK educate, 2015).

Uso en Criptografía

En criptografía, se elige un punto base G **específico** y **publicado** para utilizar con la curva $E(q)$, comentado en el apartado 3.4.3.

Se escoge un número entero **aleatorio** k como **clave privada**, y entonces el valor $P = kG$ se da a conocer como clave **pública** (nótese que la supuesta dificultad del PLDCE implica que k es difícil de deducir a partir de P).

Si María y Pedro tienen las **claves privadas** k_A y k_B, y las **claves públicas** P_A y P_B, entonces María podría calcular $k_A \times P_B = (k_A \times k_B) \times G$; y Pedro puede obtener el mismo valor dado que $k_B \times P_A = (k_B \times k_A) \times G$.

Esto permite establecer un **valor «secreto»** que tanto María como Pedro pueden calcular fácilmente, pero que es muy complicado de derivar para una tercera persona. Además, Pedro no consigue averiguar nada nuevo sobre k_A durante esta transacción, de forma que la clave de María sigue siendo privada.

La realización de las operaciones necesarias para ejecutar este sistema es más lenta que para un sistema de factorización o de logaritmo discreto módulo entero del mismo tamaño. De todas maneras, los autores de sistemas de CCE creen que el PLDCE es significativamente más complicado que los problemas de factorización o del PLD, y así se puede obtener la misma seguridad mediante longitudes de clave mucho más cortas utilizando CCE, hasta el punto de que puede resultar más rápido que, por ejemplo, RSA.

La CCE ha sido ampliamente reconocida como el algoritmo más fuerte para una determinada longitud de clave, por lo que podría resultar útil sobre enlaces que tengan requisitos muy limitados de ancho de banda.

NIST y ANSI X9 han establecido unos requisitos **mínimos** de tamaño de clave de **1024 bits** para RSA y DSA y de **160 bits** para CCE, correspondientes a un bloque simétrico de clave de 80 bits. NIST ha publicado una lista de curvas elípticas recomendadas de 5 tamaños distintos de claves (80, 112, 128, 192, 256). En general, la CCE sobre un grupo binario requiere una clave asimétrica del doble de tamaño que el correspondiente a una clave simétrica.

Certicom es la principal empresa comercial de CCE, esta organización posee 130 patentes, y ha otorgado licencias sobre tecnología a la National Security Agency (NSA) por 25 millones de dólares.

Certicom también ha patrocinado varios desafíos al algoritmo CCE. El más complejo **resuelto** hasta ahora, es una clave de **109 bits**, que fue roto por un equipo de investigadores a principios de 2003. El equipo que rompió la clave utilizó un ataque masivo en paralelo basado en el 'birthday attack', mediante más de **10000 PCs** de tipo Pentium funcionando continuamente durante **540 días**. Se estima que la longitud de clave mínima recomendada para CCE (**163 bits**) requeriría **108 veces** los recursos utilizados para resolver el problema con 109 bits (Bill Buchanan OBE, 2018; En Wikipedia, s.f.[i]).

3.4.4 Función Hash

A las funciones **resumen** también se les llama funciones **hash** o funciones digest.

Tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte en un rango de salida **finito**, normalmente cadenas de longitud fija. Es decir, la función actúa como una **proyección** del conjunto U sobre el conjunto M.

Hay que tener en cuenta que M puede ser un conjunto definido de enteros. En este caso podemos considerar que la longitud es fija si el conjunto es un rango de números de enteros ya que podemos considerar que la longitud fija es la del número con mayor número de cifras. Todos los números se pueden convertir al número especificado de cifras simplemente anteponiendo ceros.

Normalmente el conjunto U tiene un número elevado de elementos y M es un conjunto de cadenas con un número más o menos pequeño de símbolos.

La **idea** básica de un valor hash es que sirva como una **representación compacta** de la cadena de **entrada**.

Por esta razón se dice que estas funciones resumen datos del conjunto dominio como se observa en el ejemplo de la figura 3.12.

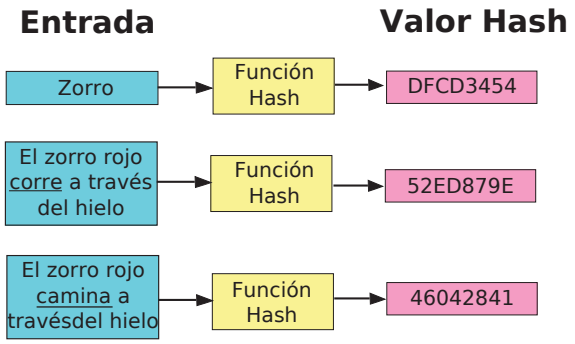


Figura 3.12 Funcionamiento de función hash (Fercufer, 2011).

Se dice que se produce una **colisión** cuando dos entradas **distintas** de la función resumen producen la **misma** salida.

De la definición de función resumen podemos decir que U, el dominio de la función, puede tener infinitos elementos. Sin embargo M, el rango de la función, tiene un número finito de elementos debido a que el tamaño de sus cadenas es fijo.

Por tanto la posibilidad de existencia de colisiones es intrínseca a la definición de función hash. Una buena función resumen es una que tiene **pocas** colisiones en el conjunto esperado de entrada. Es decir, se desea que la probabilidad de colisión sea muy baja (En Wikipedia, s.f.[o]).

Propiedades

- **Bajo costo (computacional, memoria,etc)**
- **Compresión**
- **Uniforme:** Se dice que una función resumen es uniforme cuando para una clave elegida aleatoriamente es igualmente probable tener un valor resumen determinado, independientemente de cualquier otro elemento.
- **Inyectividad y función hash perfecta:** Se dice que la función resumen es inyectiva cuando cada dato de entrada se mapea a un valor resumen diferente. En este caso se dice que la función resumen es perfecta.

  Formalización:

  $k1 \neq k2$ implica $h(k1) \neq h(k2)$

  Cuando no se cumple la propiedad de inyectividad se dice que hay colisiones. Hay una colisión cuando $k1 \neq k2$ implica $h(k1) = h(k2)$.
- **Determinista:** Una función hash se dice que es determinista cuando dada una cadena de entrada siempre devuelve el mismo valor hash.
- **Continuidad. Efecto avalancha:** Se dice que una función resumen es continua cuando una modificación minúscula (ej un bit) en la cadena de entrada ocasiona

pequeños cambios en el valor hash. En funciones resumen usadas para búsqueda normalmente se buscan funciones tan continuas como sea posible; de forma que entradas que difieran un poco deberían tener valores hash similares o iguales. Sin embargo la continuidad no es deseable para funciones resumen usadas para sumas de verificación o funciones **criptográficas**.

Seguridad

La **seguridad** proporcionada por un algoritmo **hash** es sumamente dependiente de su capacidad de producir un **único** valor para un conjunto de datos dados.

Cuando una función hash produce el mismo valor para dos conjuntos de datos distintos, entonces se dice que se ha producido una **colisión**. Una colisión aumenta la posibilidad de que un atacante pueda elaborar computacionalmente conjuntos de datos que proporcionen acceso a información segura o para alterar ficheros de datos informáticos de tal forma que no cambiara el valor hash resultante y así eludir la detección.

Una función hash **fuerte** es aquella que es **resistente** a este tipo de ataques computacionales mientras que una función hash **débil** es aquella donde existe una creencia casi certera de que se pueden **producir colisiones**. Finalmente, una función hash **quebrantada** es aquella sobre la que se **conoce** métodos computacionales para producir colisiones (En Wikipedia, s.f.[v]).

Función SHA-2 y SHA-256

SHA-2 es un conjunto de funciones hash criptográficas (SHA-224, **SHA-256**, SHA-384, SHA-512) diseñadas por la Agencia de Seguridad Nacional (NSA) y publicada en 2001 por el Instituto Nacional de Estándares y Tecnología (NIST) como un Estándar Federal de Procesamiento de la Información (FIPS). SHA significa algoritmo de hash seguro (En Wikipedia, s.f.[o]; Anders Brownworth, 2016).

- Tamaño máximo del mensaje de entrada en SHA-256: 2^{64}-1 bits.
- Tamaño de salida en SHA-256: 256 bits.

La función SHA-256 es la función Hash utilizada en la Blockchain **Bitcoin,** ver figura 3.13y tiene 2 características:

- **Inequívoca:** el hash (salida) es la huella digital de los datos de entrada. A partir del hash de una entrada digital, no se puede crear la entrada digital original.
- **Resistente a colisiones:** no es posible encontrar dos valores de entrada diferentes que resulten en la misma salida hash. Para entradas diferentes, siempre habrá salidas diferentes. Esto permite utilizar esta función para verificar la integridad de los datos comparando el "hash" (la salida de la ejecución del algoritmo) de esos datos con un valor hash ya conocido y esperado de esos datos.

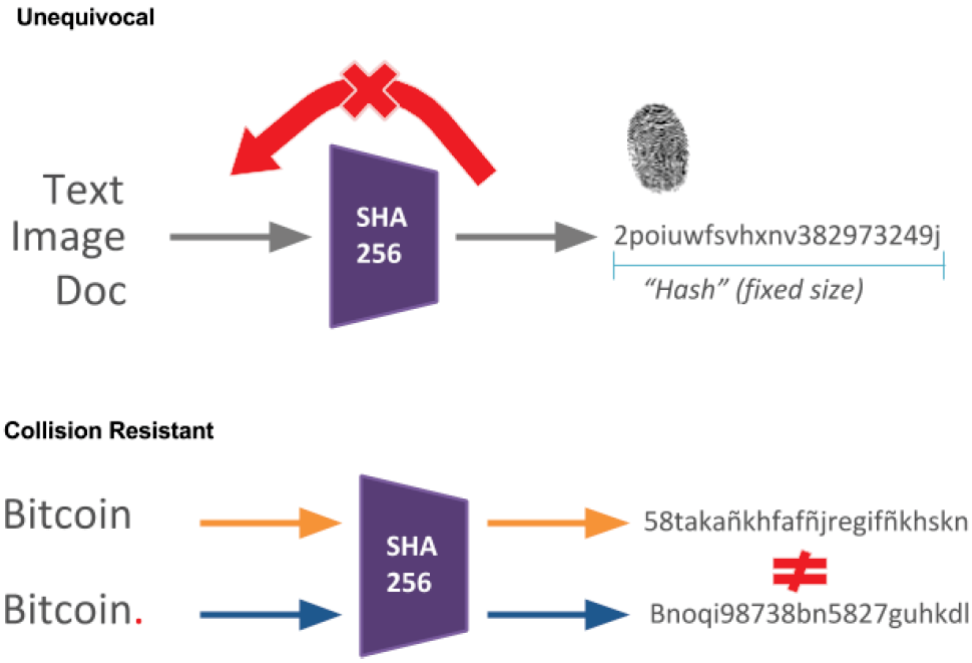


Figura 3.13 Características SHA-256 (RSK educate, 2015).

Otras funciones hash utilizadas en Blockchains son (En Wikipedia, s.f.[n]):

- **SHA-256:** Bitcoin, Bitcoin Cash, Counterparty, MazaCoin, Namecoin, NeuCoin, Nxt, Peercoin.
- **Ethash:** Ethereum, Ethereum Classic
- **Scrypt:** Auroracoin, Bitconnect, Bitcoin Gold, Coinye, Dogecoin, Gridcoin, Litecoin.
- **Equihash:** Zcash, Zcoin.
- **CryptoNote:** Monero.
- **X11:** Dash, Petro.
- **Lyra2:** Taler.

3.5 Llaves públicas y privadas, direcciones y monederos

3.5.1 Llaves públicas y privadas en Blockchain

Como dice **Andreas M. Antonopoulos**: "Son tus claves, son tus bitcoins. No son tus claves, no son tus bitcoins".

Esto resume algo **fundamental**: como la red Bitcoin utiliza criptografía para asegurar que cada uno pueda utilizar sus bitcoins de forma segura, mantener segura y secreta la clave privada es la única forma de mantener seguros los bitcoins.

Aquel que conozca esa clave privada, puede utilizar esos bitcoins ya que el sistema no tiene forma de distinguir entre el dueño y el "falso" dueño. La clave privada es la "**prueba de identidad**" necesaria para que la red acepte cualquier movimiento de esos bitcoins.

Si se perdiera la clave privada, esos bitcoins quedarían inmovilizados por el resto de la eternidad.

El motivo es parte del típico dilema de la seguridad vs. la facilidad de uso. En este caso, bien vale el costo de perder un poco de usabilidad y no poder recuperar claves porque el beneficio es tener un sistema de máxima **seguridad** y prácticamente **inviolable** (Gabriel Montes, 2017).

¿Qué son el par de llaves pública-privada?

1. **Claves privadas**

   Como se ha comentado en los apartados 3.4.2 y 3.4.3, la clave privada es un número entero que deber ser conocido solo por el usuario/dueño de esa clave, que en Bitcoin es un número de 256 bits en un rango entre $[0 - P(2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1)]$ ya que P es el límite del plano finito de la curva $secp256k1$, por seguridad debe ser generado de forma aleatoria (aunque no es obligatorio) y utilizando la criptografía de curvas elípticas para generar la clave pública.

   A partir de una clave privada es posible generar miles de claves públicas, ver apartado 3.5.1, y la clave privada permite firmar transacciones en la Blockchain (**firma digital**) demostrando ser el propietario de esas criptodivisas sin necesidad de desvelar la clave privada.

2. **Claves públicas y direcciones**

   Como se ha comentado en los apartados 3.4.2 y 3.4.3, la clave pública será un número generado matemáticamente a partir de la clave privada, pero es imposible calcular la clave privada a partir de la pública, que se puede compartir libremente para recibir mensajes, en el caso de las Blockchains transacciones.

   Las claves públicas, pueden tener 256 o 512 bits dependiendo si están o no comprimidas, a los cuales se les suma un byte adicional como prefijo indicativo del tipo de clave pública en cuestión. Por lo tanto, las claves públicas son siempre de 33 o 65 bytes.

   Para ahorrar espacio en la blockchain y facilitarle la vida a los usuarios, existe una alternativa adicional a utilizar una clave pública de ese tamaño: crear una "**dirección**" al reducir esa clave pública a 160 bits a partir de aplicar **SHA256** y luego **RIPEMD-160** sobre la clave pública. Estas direcciones por la forma de codificar los 160 bits utilizando **Base58Check** pueden tener entre 26 y 35 caracteres en total y comienzan con "1" en Bitcoin.

   Por tanto la **dirección pública** no es la clave pública y la clave pública **no** se puede conocer/generar a partir de una dirección pública, la clave pública solo es posible conocerla al generarla a partir de la clave privada o en una transacción firmada donde al enviar bitcoins desde esa dirección se revela la clave pública, si tienes una dirección solo para recibir transacciones, no es posible conocer la llave pública.

   Es importante destacar que si bien de la clave privada se deriva la pública, y de ésta la dirección, no es posible realizar el camino inverso. De ahí la seguridad de

que haciendo pública la clave pública o la dirección, no es posible para un tercero conocer o encontrar la clave privada que las generó (Gabriel Montes, 2017; Bit2Me, s.f.[a]; Andreas M. Antonopoulos, aantonop, 2014).

Monederos o Wallets

Los **Monederos** de criptomonedas (o **Wallets** en inglés), deberían llamarse "**Llaveros**" ya que las criptomonedas están siempre en la Blockchain, y los Wallets lo único que contienen son las **llaves privadas** del dueño de esas criptomonedas.

Estos Wallets son un **software** que almacena las llaves privadas, permite firmar transacciones y enviarlas a la Blockchain, también existen Wallets **físicas**, de tipo Hardware (que no es un móvil o un ordenador) que permiten firmar las operaciones y permiten "Cold-Storage" (Wallets que no están conectadas a Internet), o algo tan simple como imprimir la llave privada en un papel o grabarlo en una roca; estos dos últimos métodos, almacenan la llave privada, pero no pueden firmar transacciones ni comunicarse con la Blockchain, son Cold Storage, y para usar esa clave privada en el futuro es necesario cargarla en un Wallet con conexión a Internet.

¿Cómo se genera el par de llaves pública-privada y las direcciones públicas correspondientes?

Bitcoin utiliza un algoritmo llamado ECDSA (**Elliptic Curve Digital Secure Algorithm**) para derivar de una clave privada, la clave pública correspondiente y así poder validar las firmas digitales hechas con esa clave privada.

Cualquier número de 256 bits o 32 bytes puede ser una clave privada válida. Por lo tanto, solo basta con calcular el SHA-256 de cualquier cadena de caracteres para obtener una clave privada válida (Gabriel Montes, 2017; Andreas M. Antonopoulos, aantonop, 2014; RSK educate, 2015).

Generación de llaves no deterministas (se usa una fuente aleatoria):

1. Es necesario partir de un número, puede ser cualquier número, pero es recomendable tener una fuente confiable de aleatoriedad y obtener un número aleatorio, para ello es necesaria una fuente de **entropía**, un generador de números aleatorio que haya sido propiamente configurado y con propiedades específicas, criptográficamente seguro y correctamente inicializado (seeded) con una buena fuente de datos aleatorios/entropía (ej. fuentes cósmicas), **Cryptographically Secure random Number Generator (CSRNG)**.

2. Alimentar un algoritmo de **SHA-256** con este número aleatorio (tamaño máximo $2^{64} - 1$ bits y recomendable > 256 bits). El resultado es un número aleatorio de 256 bits, que será la Llave Privada/Private key y que debido a las propiedades del SHA-256 se puede volver a generar si se introduce de nuevo el mismo número aleatorio.

   a) Ejemplo de Clave Privada:
   0C28FCA386C7A227600B2FE50B7CAE11EC86D3BF1FBE471BE89827E19D72AA1D

3. **Wallet Import Format (WIF):** es una forma de codificar una clave ECDSA privada para facilitar la copia (bitcoin.it, 2017).
   Convierte la Clave Privada en un formato genérico usando *"Codificación Base58Check"*. Debido a esto todas las Claves privadas de la Blockchain principal (mainnet) de Bitcoin empiezan por **"5"** y las claves privadas de la Blockchain de prueba (testnet) empiezan por "n" o "m", en otras Blockchains las claves privadas empiezan por otro número o letra. Es posible volver del formato WIF al formato SHA-256.

   a) Ejemplo de la **Clave Privada** en formato WIF:
   5HueCGU8rMjxEXxiPuD5BDku4MkFqeZyd4dZ1jvhTVqvbTLvyTJ

4. Esta clave privada (k), se hace pasar por la curva elíptica, para Bitcoin $secp256k1$, y se obtiene la clave pública (P) donde $P = k * G$. La clave pública es ahora un punto de gráfica por tanto $P = (x_p, y_p)$ donde x_p e y_p son dos números de 32 bytes cada uno. La representación hexadecimal de la clave pública (P_k) es $04\,x_p\,y_p$, correspondiente a un número hexadecimal de 130 caracteres.

5. Este número hexadecimal (P_k), se hace pasar por el algoritmo **RIPEMD160**, una función de compresión/hash cuya salida es un número de 160 bits, y de nuevo por el algoritmo **SHA256,** como resultado se obtiene una **dirección** Bitcoin pública cuya codificación en Base58check empieza siempre por **"1"**.

   El número hexadecimal (P_k), está formado por $04\,x_p\,y_p$, que pertenecen a una curva elíptica, que es simétrica respecto al eje X, por tanto, conocida x_p y el signo/lado de la curva es posible conocer y_p y ahorrar 32 bytes de memoria. P_k, se puede comprimir como $02\,x_p$ para cuando P_k tenga signo negativo/se encuentre en el lado "inferior" de la curva o $03\,x_p$para el signo positivo de y_p.

   a) Ejemplo de **Clave Pública** sin comprimir (130 caracteres):
   04D0DE0AAEAEFAD02B8BDC8A01A1B8B
   11C696BD3D66A2C5F10780D95B7DF42645C
   D85228A6FB29940E858E7E55842AE2BD115
   D1ED7CC0E82D934E929C97648CB0A

   b) Ejemplo de **Clave Pública** comprimida (66 caracteres):
   02D0DE0AAEAEFAD02B8BDC8A01A1B8B
   11C696BD3D66A2C5F10780D95B7DF42645C

   c) Ejemplo de la **Dirección** Bitcoin Pública en formato WIF generada de la Clave Privada inicial:
   1GAehh7TsJAHuUAeKZcXf5CnwuGuGgyX2S

   d) Ejemplo de la **Dirección** Bitcoin Pública en formato WIF a partir de la Clave Pública comprimida generada de la Clave Privada inicial:
   1LoVGDgRs9hTfTNJNuXKSpywcbdvwRXpmK

6. La clave privada siempre genera de forma **determinista** la misma clave pública que siempre genera la misma dirección pública, no es necesario por tanto **almacenar** ninguna clave o dirección pública, se pueden **regenerar** infinitas veces a partir de la clave privada.

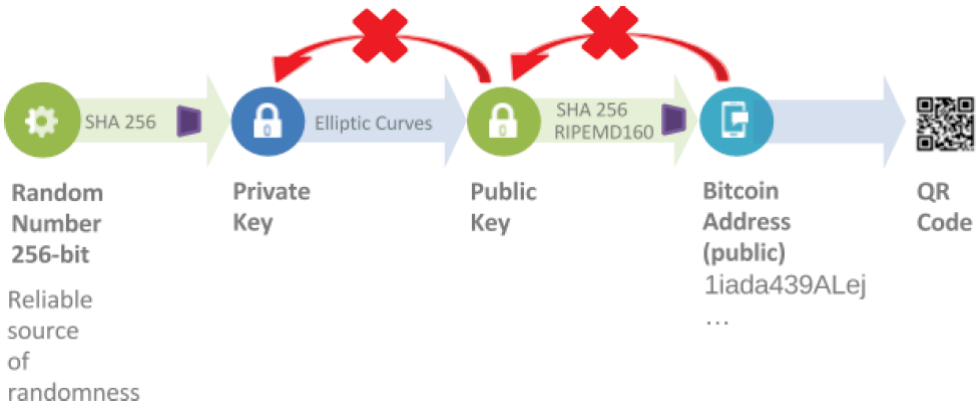


Figura 3.14 Generar una dirección pública en Bitcoin (RSK educate, 2015).

Claves derivadas

El protocolo Bitcoin genera direcciones para el "cambio" que "sobra" tras una transacción, se verá en el apartado 3.7, y además para disminuir riesgos y aumentar la privacidad de los usuarios es recomendable usar una dirección **diferente** para cada transacción y no **reutilizar** direcciones, por ello el método de generación de claves visto anteriormente requiere gestionar un gran número de llaves privadas, haciendo backup de esas llaves después de cada transacción, esto es una **desventaja**.

Aparecen dos soluciones:

1. **Monederos deterministas basados en "cadenas"**

   La idea principal de este monedero es, a partir del número aleatorio obtenido en el paso 1 del método anterior, se genera una **"Seed/Semilla"** hexadecimal de 128 bits, que será el **generador** de todas las direcciones futuras.

   Una función matemática de sentido único, calcula k_0,k_1,k_2,...,k_n a partir de la Seed, no es posible predecir las k_n sin conocer la Seed, y no es posible a partir de k_n conocer la Seed, y no es posible conociendo k_1 conocer k_2 ni viceversa, ver figura 3.15.

   Conociendo la Seed se pueden **regenerar** todas las k_n, por tanto no es necesario almacenar todas las direcciones, solo hay que guardar y mantener en secreto la Seed.

   A partir del esquema **BIP39**, existe una forma sencilla de recordar o almacenar de manera segura esas semillas de 128 bits: a través de **12 palabras**.

   Este esquema define un diccionario de 2048 palabras. Cada palabra tiene asignados 11 bits (del 0x000 al 0x7FF) y fue seleccionada de forma tal que sea sencilla y que sus 4 primeras letras sean distintas de cualquier otra palabra del diccionario, evitando confusiones entre palabras similares.

   Sumando 12 de estas palabras se obtienen entonces **132 bits**, de los cuales 4 son utilizados para detección de errores, quedando los **128 bits** restantes para formar la semilla (Andreas M. Antonopoulos, aantonop, 2014; Gabriel Montes, 2017).

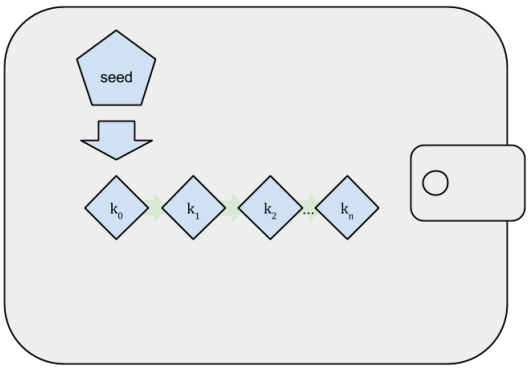


Figura 3.15 Monedero Determinista tipo Cadena (Nick Adams, 2017).

2. **Monederos deterministas basados en "árboles"**

   Al esquema **BIP32** se le llama **Hierarchical Deterministic Wallet**, o monedero jerárquico determinista, porque genera una jerarquía en forma de **árbol**, de *n* ramas, donde cada rama tiene 32 bits, eso implica que existen 4 mil millones de ramas por cada rama principal, con este método se alcanzan números astronómicos de posibles pares de llaves, ver figura 3.16.

   El método es similar al modelo basado en cadenas, parte de una Semilla, que genera una **Master key/Llave maestra**, que genera una serie de llaves privadas, que pueden tener subllaves privadas, que pueden tener subllaves privadas, etc.

   Este método genera pares de subllaves pública-privada, pero el gran potencial es que permite generar a partir de la Master key, **subllaves públicas**, sin necesidad de **conocer** la llave privada, y todas las llaves se pueden regenerar en otro Wallet simplemente conociendo las **12** palabras de la **Semilla** (Andreas M. Antonopoulos, aantonop, 2014; Gabriel Montes, 2017).

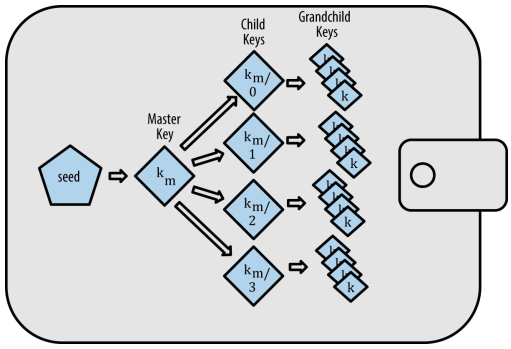


Figura 3.16 Monedero Determinista tipo Árbol (Nick Adams, 2017).

.

3.6 Nodos y Mineros

3.6.1 Nodos

Un **nodo** de red es un punto en el que se puede **crear**, **recibir** o **transmitir** un mensaje, es un dispositivo que se conecta a la red Blockchain y utiliza el protocolo peer-to-peer (**P2P**, por sus siglas en inglés) que permite que los nodos se comuniquen entre sí dentro de la red, así como difundir información sobre transacciones y bloques, al hacerlo, garantizan la **integridad** del sistema.

Un nodo con mal comportamiento o que intenta propagar información incorrecta es reconocido rápidamente por los nodos honestos y se desconecta de la red.

Existen diferentes tipos de nodos: nodos completos, supernodos, nodos mineros y clientes SPV (bitcoinwiki.org, 2019b; En Wikipedia, s.f.[r]; binance.vision, 2018).

3.6.2 Nodos completos

Los nodos que hacen cumplir plenamente todas las **reglas** de las Blockchains se llaman **nodos completos**. Esto significa que están completamente sincronizados con la red de la blockchain, es decir, almacenan el Ledger completo de la blockchain en una unidad de disco duro hasta la fecha.

Los nodos completos son los que realmente admiten y proporcionan **seguridad** a la Blockchain y son **indispensables** para la red. Estos nodos también se conocen como nodos de validación total a medida que se involucran en el proceso de verificar las transacciones y los bloques con respecto a las reglas de consenso del sistema. Los nodos completos también pueden transmitir nuevas transacciones y bloques a la Blockchain.

Cabe señalar que la propiedad de un nodo completo y su participación en la validación de la transacción no es minería (la búsqueda de nuevos bloques en la red para obtener una recompensa), es decir, su presencia en una Blockchain no le garantizará ninguna ganancia. Ejecutar un nodo completo es la forma más **segura** de descargar transacciones con el uso de criptodivisas. Además, la **estabilidad** de todo el funcionamiento del sistema depende de la estabilidad de funcionamiento de los nodos completos.

A pesar del hecho de que la ejecución de un nodo de validación completo no proporciona recompensas, se recomienda encarecidamente porque proporciona **confianza**, **seguridad** y **privacidad** a los usuarios.

Los nodos completos aseguran que se sigan las **reglas**. **Protegen** la Blockchain contra ataques y fraudes (como el doble gasto). Además, un nodo completo no necesita confiar en otros y le permite al usuario tener el control total de sus transacciones.

Operación de nodos completos

Los nodos completos descargan cada **bloque** y **transacción** y los comprueban contra las **reglas** principales de consenso.

Si una transacción o bloque **viola** las reglas de consenso, un nodo lo **rechaza** por completo, incluso si todos los demás nodos de la red lo consideran válido. Además, durante un tiempo dejará de comunicar con la fuente de la transacción que ha intentado descargarlo mientras viola las reglas de consenso.

Es una de las características más importantes de los nodos **completos**: hacen lo **correcto** sin importar qué, así garantiza un alto nivel de **seguridad** de las transacciones.

3.6.3 Supernodos o nodos de escucha

Esencialmente, un nodo de escucha o **supernodo** es un nodo completo que es **públicamente visible**. Se comunica y proporciona información a cualquier otro nodo que decida establecer una conexión con él. Por lo tanto, un supernodo es básicamente un punto de **redistribución** que puede actuar como fuente de datos y como puente de comunicación.

Un supernodo confiable generalmente se ejecuta las 24 horas del día, los 7 días de la semana y tiene varias **conexiones** establecidas, **transmitiendo** el historial de la Blockchain y los datos de las transacciones a **múltiples** nodos en todo el mundo.

Por esa razón, un supernodo probablemente requerirá más **poder** de cómputo y una mejor **conexión** a Internet en comparación con un nodo completo que está **oculto**.

3.6.4 Nodos mineros

Un **nodo minero** es un nodo completo con capacidad de **minar** bloques, mientras que los mineros tienen que invertir en **hardware** y **software** de minería **caros** y especializados, cualquiera puede ejecutar un nodo de validación completo.

Antes de intentar minar un bloque, un minero debe **recopilar** las **transacciones** pendientes que previamente fueron aceptadas como **válidas** por los nodos completos.

A continuación, el minero crea un bloque candidato (con un grupo de transacciones) y trata de minar ese bloque.

Si el minero logra encontrar una solución **válida** para ese bloque, lo **transmite** a la red y los otros nodos completos verificarán la **validez** del bloque.

Por lo tanto, las **reglas de consenso** están determinadas y aseguradas por la **red distribuida** de nodos de validación y no por los mineros.

Pool de minería

Las **mining pools** son las agrupaciones de equipos mineros que se unen en una red determinada para compartir y ampliar su capacidad o poder de procesamiento, agilizando así su capacidad para resolver una cadena de bloques criptográficos y así tener mayores probabilidades de tener éxito.

Las ganancias se distribuyen de forma proporcional al hashrate, o poder computacional que aporta su máquina en un momento determinado (crypto-economy.net, 2017).

3.6.5 Nodos livianos/ligeros o SPV

También conocidos como clientes de Verificación de Pago Simplificada (**SPV**), los clientes ligeros son los que hacen **uso** de la **red** pero en realidad no actúan como un nodo completo.

Los clientes SPV no contribuyen a la seguridad de la red porque no mantienen una copia de la Blockchain y no participan en el proceso de verificación y validación de las transacciones.

Es el método mediante el cual un **usuario** puede **verificar** si algunas transacciones se incluyeron o no en un bloque, **sin** tener que **descargar** los datos del bloque completo, los clientes SPV se basan en la información proporcionada por otros nodos completos (supernodos).

Los clientes ligeros trabajan como puntos finales de comunicación y son utilizados por muchas carteras de criptomonedas.

No son tan independientes como los nodos completos. Por esta razón, se les podría engañar temporalmente para que acepten una transacción o un bloque que no es realmente válido.

Ejecutar un nodo **completo** es la mejor forma en que puede usar Blockchain de la manera más segura posible (binance.vision, 2018; bitcoinwiki.org, 2019b; En Wikipedia, s.f.[r]).

3.7 Transacciones, Bloques y Blockchain

En este apartado 3.7, se estudia el la estructura básica de la Blockchain, que está compuesta por Bloques relacionados que contienen las Transacciones de los usuarios.

3.7.1 Transacciones

Componentes

Una transacción se compone de 5 elementos clave, ver figura 3.17:

1. **Identificador de Transacción (TX ID):** identificador único (32 bytes) de cada transacción, obtenido de pasar 2 veces la información de la transacción por la función SHA-256 (learnmeabitcoin, s.f.).
2. **Entrada (Origen):** dirección o direcciones públicas de origen que poseen los Tokens.
3. **Cantidad**: cantidad de Tokens que se enviarán en la transacción.
4. **Salida (Destino):** dirección o direcciones públicas de destino.
5. **Metadatos (Opcional):** los metadatos o mensajes tienen un tamaño máximo de 80 bytes y se almacenan en“OP_RETURN”, campo utilizado para transmitir cualquier tipo de información, de la transacción.

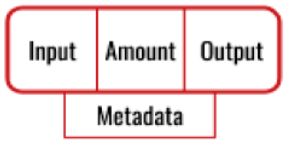


Figura 3.17 Componentes de una Transacción (RSK educate, 2015).

¿Qué son las transacciones en Blockchain?

Una transacción es la operación **básica** en el sistema Blockchain. Una transacción mueve los tokens entre una o más **entradas** y **salidas**.

Cada entrada es una transacción anterior y una dirección que proporciona tokens. Cada salida es una dirección que recibe tokens, junto con la cantidad de tokens que van a esa dirección.

Una transacción está compuesta por entradas y salidas de transacciones no gastadas o **UTXO** (unspent transaction outputs). Las entradas de transacciones también son UTXOs

de una transacción anterior, por lo tanto, una **UTXO** es el componente **fundamental** de una transacción.

Para que un Wallet muestre su "saldo", debe buscar en la Blockchain los UTXO que controla su clave privada, sumar los valores de los UTXO y muestra el saldo final.

Cuando desea gastar un token, el Wallet comprobará si tiene suficientes UTXO que sumen un token. Si tiene suficiente, el Wallet creará otra transacción con estos UTXO como entradas.

El UTXO es una unidad de valor **discreta** e **indivisible**.

El diagrama 3.18, muestra la Transacción C.

En esta transacción, se toman 0.005 BTC de una dirección en la Transacción A, y 0.003 BTC de una dirección en la Transacción B.

Las flechas son referencias a las salidas anteriores, por lo que son hacia atrás al flujo de bitcoins, este tipo de transacciones forman una estructura de **Árbol de Merkle**, ver apartado 3.7.2.

Para las salidas, 0.003 BTC se envían a la primera dirección y 0.004 BTC se envían a la segunda dirección. El sobrante de 0.001 BTC va al minero del bloque como un comisión.

El 0.015 BTC en la otra salida de la Transacción A no se envía en la Transacción C.

Cada entrada UTXO es **indivisible**, debe **gastarse** por completo en una transacción. Si una dirección recibe 100 BTC en una transacción y solo desea gastar 1 BTC, la transacción debe gastar los 100 BTC.

La solución es usar una segunda salida para el cambio, que devuelve los 99 BTC restantes a una **dirección propia**.

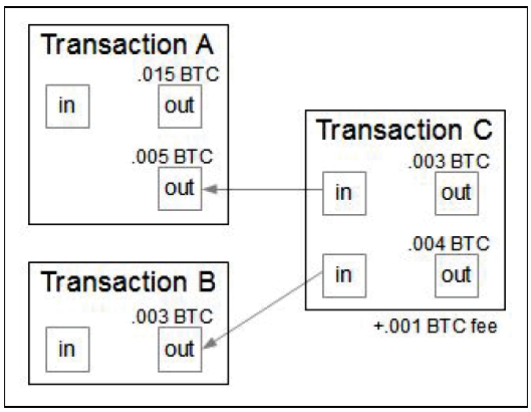


Figura 3.18 Ejemplo de una Transacción (RSK educate, 2015).

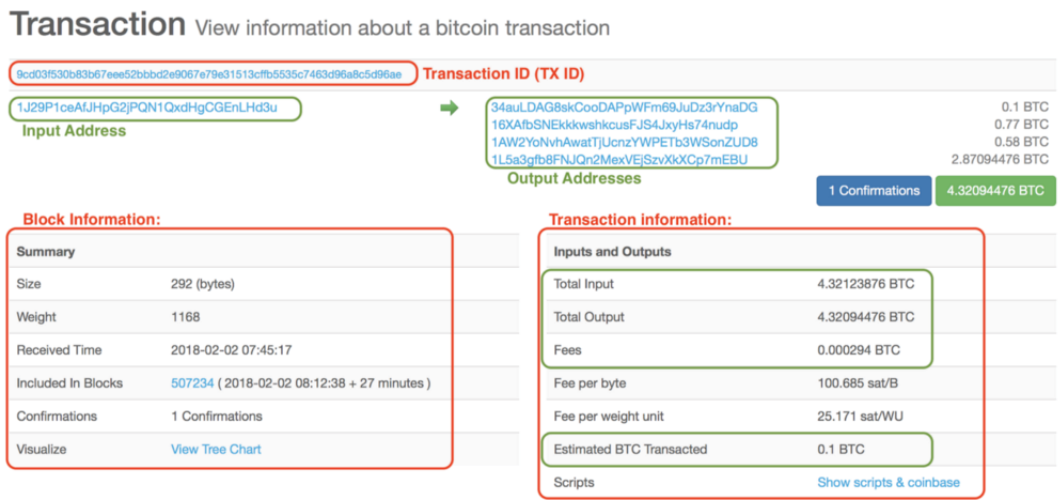


Figura 3.19 Transacción Real de Bitcoin (sheinix, 2018).

En la figura 3.20, se realiza la siguiente transacción, se cogen las UTXO suficientes para alcanzar la cantidad deseada; para alcanzar 0.3 BTC ha tenido que seleccionar 3 UTXO de 0.12 BTC cada una, siendo el resultado 0.36 BTC, los otros 0.06 BTC restante se envían a una dirección propia indicándola como salida junto a la dirección a la que se le quería mandar 0.3 BTC.

La dirección propia donde se mandan los 0.06 BTC restantes puede ser la misma dirección que cualquiera asociada a las entradas o una nueva. A esto se le llama dirección de **cambio** o dirección de **retorno**.

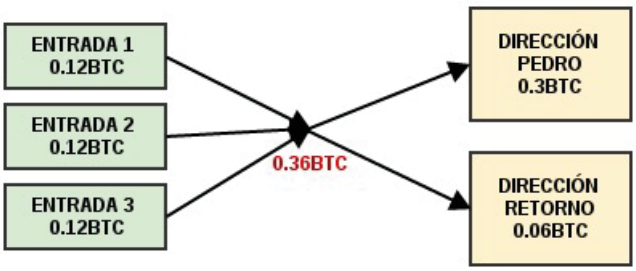


Figura 3.20 Transacción con Retorno (Bit2Me, s.f.[c]).

Es importante entender también que en una misma transacción pueden existir tantas entradas de una misma dirección o de varias como se desee. Lo mismo ocurre con las salidas. Esto permite hacer en una misma transacción **múltiples** envíos a personas diferentes con un sólo **pago** de comisiones a los mineros. Esta funcionalidad la explotan algunos monederos/wallets para ahorrar costes.

Las transacciones también pueden incluir **tasas/comisiones**. La forma que tiene internamente el protocolo de recompensar a los mineros viene de los fondos que no se asignen a ninguna dirección. Todos los bitcoins restantes en una transacción que **no se asignen** a ninguna dirección se los queda el minero que mine el bloque con la transacción dentro y son imposibles de recuperar.

Si sobran bitcoins después de sumar las entradas y restar las salidas, el resto es una **comisión** pagada al minero. La comisión no es estrictamente necesaria, pero las transacciones sin una tarifa serán de baja **prioridad** para los mineros y pueden no ser procesadas en días o incluso ser descartadas por completo. Las tarifas son **bajas** pero no triviales (sheinix, 2018; Bit2Me, s.f.[c]; RSK educate, 2015).

Firmar Transacciones

Una vez creada la transacción es necesario firmarla con la clave privada para validarla y demostrar la propiedad de esos Tokens.

El diagrama 3.21 ofrece una vista simplificada de cómo las transacciones se firman y vinculan entre sí.

Considere la transacción intermedia, transacción de la dirección B a la dirección C. El contenido de la transacción (incluido el **hash** de la transacción **anterior**) está marcado y firmado con la **clave privada** de B. Además, la clave pública de B está incluida en la transacción.

Realizando varias operaciones, cualquiera puede verificar que la transacción esté **autorizada** por B.

Primero, la **clave pública** de B debe corresponder a la **dirección** de B en la transacción anterior, lo que demuestra que la clave pública es **válida**, la dirección se puede derivar fácilmente de la clave pública, como se explicó anteriormente.

A continuación, la firma de B, generada con la clave privada, de la transacción se puede **verificar** utilizando la **clave pública** de B en la transacción.

Estos pasos garantizan que la transacción sea válida y autorizada por B.

Se observa que la clave pública de B no se hace pública hasta que se usa en una transacción.

Con este sistema, los tokens pasan de una dirección a otra a través de una cadena de transacciones. Cada paso en la cadena se puede verificar para asegurar que se estén enviando de manera válida. Tenga en cuenta que las transacciones pueden tener múltiples entradas y salidas en general, por lo que la cadena se ramifica en un árbol.

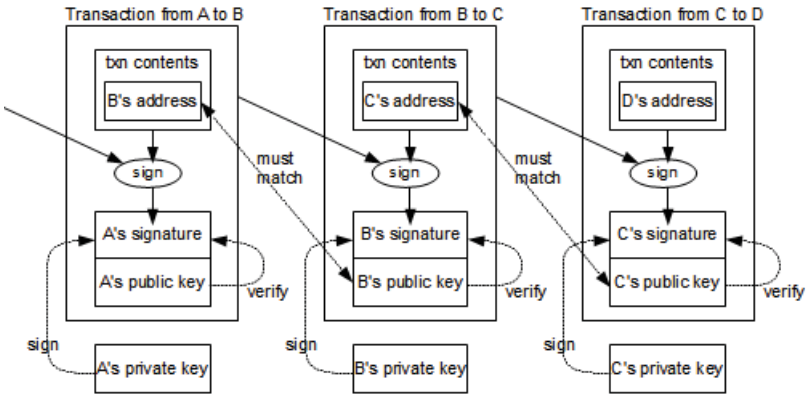


Figura 3.21 Firma de una Transacción (Shirriff, 2014).

3.7.2 Bloques

Todos los datos son grabados permanentemente en la red Blockchain a través de **bloques**, un bloque es un concepto pensado para **optimizar** el proceso de **validación** de las transacciones que se realizan.

En Blockchain hay decenas de transacciones por segundo, validar de manera individual cada una de estas operaciones, sería **inviable** y sería un proceso largo y **tedioso**. Así que se ideó la creación de los bloques, que permiten acortar la cadena hash y hacerla más manejable y por lo tanto, permite ser más **eficiente** (Bit2Me, s.f.[b]).

Cada bloque contiene un conjunto de las últimas transacciones, un nonce (número aleatorio), y el hash de la secuencia anterior.

Un bloque se considera "resuelto" (publicado y considerado válido por el resto de nodos), cuando el hash SHA-256 de todo el bloque cumpla el objetivo de dificultad actual, se estudia en profundidad en el apartado de minería, 4.2.1.

Las transacciones firmadas se transmiten a la red por el remitente, y todos los participantes en la generación de los tokens las recogen y agregan al bloque en el que están trabajando.

La primera transacción en el bloque es especial, la **Coinbase**, crea nuevos tokens para el minero que lo genera y recoge las comisiones, el resto de nodos sólo acepta el bloque si la operación es de la cantidad correcta. En Bitcoin, el número de bitcoins generado por bloque comienza en 50 y se irá dividiendo entre dos (“Halving”) cada 210.000 bloques (unos cuatro años).

La red Bitcoin trata de crear **seis bloques por hora**. Cada 2016 bloques (alrededor de dos semanas), todos los clientes Bitcoin comparan el número real creado con este objetivo y modifican el objetivo por el porcentaje que ha variado. Esto aumenta (o disminuye) la **dificultad** de generación de bloques de la red (bitcoin.it, 2011a).

Estructura Bloque

Cada bloque es un grupo de muchas transacciones y cada uno contiene información específica. El campo principal de un bloque es el **encabezado/cabecera** y el secundario es el cuerpo que contiene las transacciones. El encabezado contiene estos campos (RSK educate, 2015), y está representado en la figura 3.22:

- **Versión:** un número de versión para comprobar actualizaciones de software / protocolo.
- **Hash del bloque anterior:** una referencia al hash del bloque anterior (padre) en la cadena.
- **Raíz del árbol Merkle:** un hash del Árbol Merkle de las transacciones del bloque.
- **Marca de tiempo:** la fecha aproximada de creación del bloque (segundos Unix Epoch).
- **Objetivo de dificultad:** el objetivo de dificultad del algoritmo PoW para este bloque.
- **Nonce:** un contador usado para el algoritmo de PoW, equivale al resultado buscado del algoritmo.

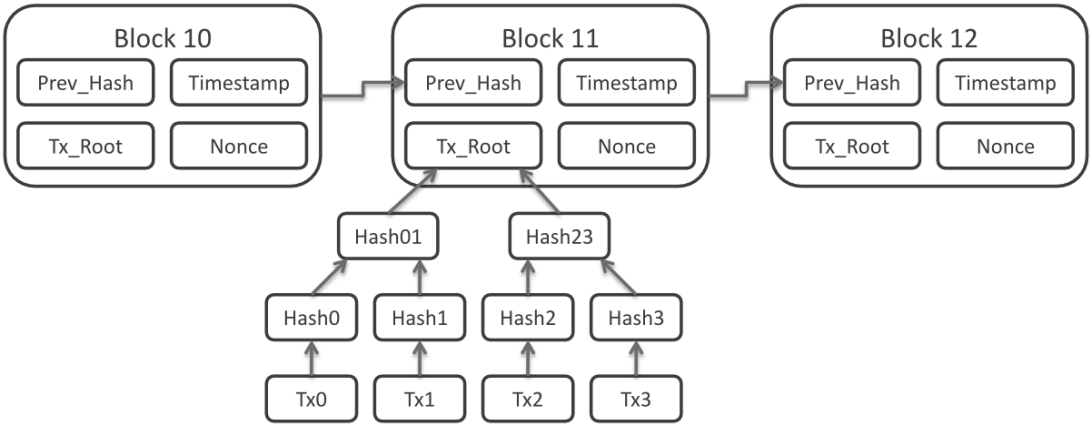


Figura 3.22 Cabecera de un Bloque (Wander, 2013).

Árboles Merkle

Cada bloque en la Blockchain contiene un resumen de todas las transacciones contenidas en el mismo, utilizando un **árbol Merkle**. Un árbol Merkle es una estructura de datos que se utiliza para resumir y verificar de manera eficiente la integridad de grandes conjuntos de datos, son árboles binarios que contienen hashes criptográficos.

Los árboles Merkle producen una **huella digital global** de todo el conjunto de transacciones, se crean mediante hash recursivo de **pares** de elementos o transacciones hasta que solo queda un hash, llamado **raíz de Merkle**. El algoritmo hash criptográfico utilizado en los árboles Merkle de Bitcoin es SHA256 aplicado dos veces, también conocido como doble SHA256.

Cuando N elementos de datos se incluyen y se resumen en un árbol Merkle, se puede verificar si un elemento en particular está incluido en el árbol mediante un número máximo de cálculos de $2log_2(N)$, que proporciona una manera muy **eficiente** de **verificar** si una transacción está incluida en un bloque o no.

El árbol Merkle está construido de abajo hacia arriba. En la Figura 3.22, comenzamos con cuatro **transacciones**; denotados como $Tx0, Tx1, Tx2, Tx3$. Estas transacciones no se almacenan en el árbol Merkle, sino que se copian sus datos y el **hash** resultante se **almacena** en cada nodo de "hoja" como $Hash0, Hash2$ y $Hash3$.

El hash raíz de Merkle de **32 bytes** se almacena en el **encabezado** del bloque y resume todos los datos de las cuatro transacciones. Usando este método, el árbol Merkle puede resumir o **reducir** cualquier número de transacciones en el bloque a tan solo un número de 32 bytes.

En los intentos de averiguar si una transacción específica está incluida en un bloque, se puede descubrir fácilmente una ruta de autenticación o camino de Merkle, que conecta la transacción específica a la raíz del árbol. Para esto solo se necesitaría producir una cantidad de hashes dado por la función:

$2log_2(N)$

Esta cantidad de "ejecuciones" de hash, son la cantidad **mínima** que permitiría encontrar una conexión entre una transacción dada y la raíz del árbol de Merkle.

Esto es importante ya que el logaritmo en base 2 de un número aumenta mucho más lentamente que el número. En consecuencia, esto significa que un nodo simplemente necesita

producir rutas de 10 o 12 hashes (320 bytes a 384 bytes) para probar una sola transacción dentro de un árbol que posea más de mil transacciones en un bloque. La eficiencia de un árbol de Merkle se hace más notoria a medida que la **escala aumenta**.

Mientras el tamaño del bloque puede variar, desde 4kB con 16 transacciones a 16MB con 65535 transacciones. Comparando esto con el "camino de Merkle" requerido para validar la inclusión de una transacción en un bloque, éste último crece desde 128 bytes a tan sólo 512 bytes.

Utilizando árboles de Merkle, un nodo solo necesita descargar los encabezados de los bloques (esto sólo ocupa 80 bytes por bloque) y así puede conocer si una transacción está dentro de un determinado bloque o no.

Esto permite a los nodos ligeros/SPV el ahorro en el espacio, ya que con solo almacenar unos pocos megabytes evitan tener que descargar la Blockchain completa y les permite verificar transacciones (Toledo, 2018).

3.7.3 Blockchain

Blockchain es la estructura de datos resultado de la configuración de los Bloques descrita en el apartado 3.7.2, uno de los campos del encabezado es el **"hash del bloque anterior"**. Esta es la forma en que los bloques están conectado como se observa en la Figura 3.23.

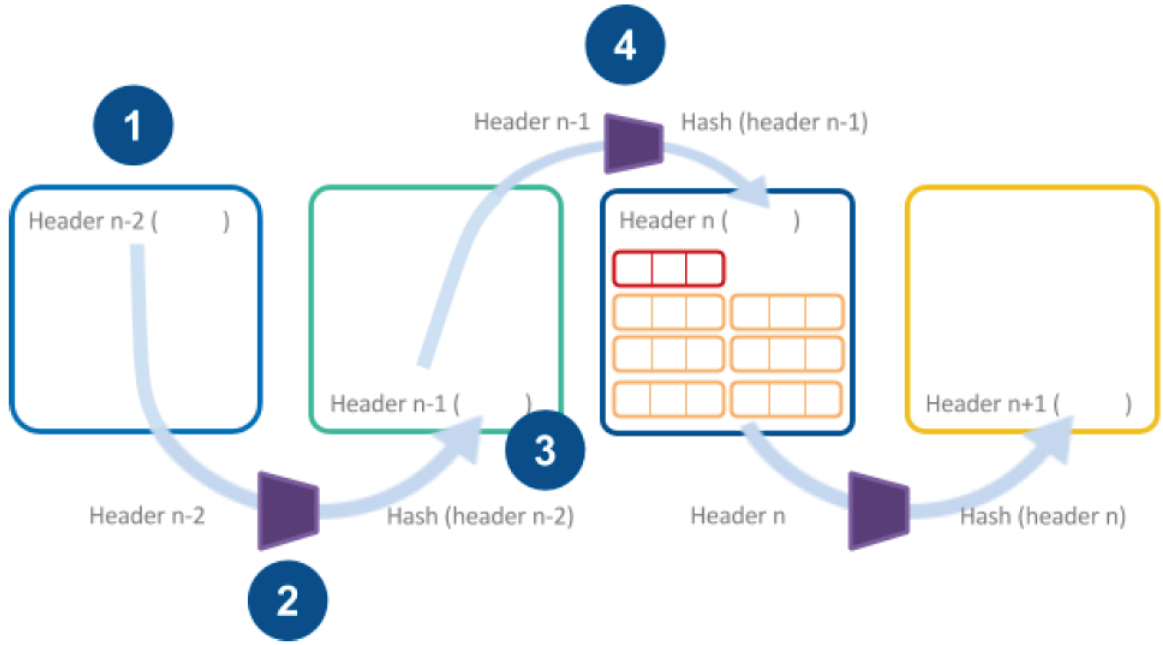


Figura 3.23 Conexión Bloques (RSK educate, 2015).

1. Se empieza desde el encabezado del bloque $n-2$.
2. Se aplica un algoritmo de hash SHA256 al encabezado $n-2$, obteniendo el $hash(encabezado\,n-2)$.

3. El $hash(encabezado\, n-2)$ se introduce en el encabezado $n-1$ (en el campo "Hash Bloque Anterior").

4. Se aplica un algoritmo de hash SHA256 al encabezado $n-1$, obteniendo el $hash(encabezado\, n-1)$ y luego se introduce en el encabezado n, y así sucesivamente.

Se obtiene una cadena de bloques, un conjunto de bloques que contienen información que están conectados entre sí con la técnica criptográfica llamada hash (RSK educate, 2015).

Si un nodo **malicioso** modifica una transacción de un bloque antiguo, modifica a su vez el árbol Merkle de ese bloque, y modifica el **Hash** del encabezado de ese bloque, el bloque siguiente tendrá en su cabecera un Hash diferente al Hash del Bloque anterior, se habrá roto la **"cadena"** y por tanto esa modificación será rechazada por la red de nodos completos.

La única forma de que esta modificación fuera aceptada sería que un nodo minero, modificara ese bloque, al modificarlo cambiaría su Hash y debería minar el siguiente bloque con el Hash nuevo, eso modificaría el Hash del siguiente Bloque y así sucesivamente hasta minar de nuevo todos los Bloques desde el modificado hasta llegar al Bloque actual de la red, para realizar esta tarea de forma exitosa es necesario controlar un 51 % del **poder de cómputo** total de la red durante mucho tiempo, en Bitcoin es prácticamente **imposible**, en Blockchains pequeñas con menos nodos y mineros sí se han dado casos de **ataques 51 %**.

Junto con la tecnología de llaves público-privadas, en esta estructura de bloques unidos/referenciados y base de datos distribuida, radica la **seguridad** de la Blockchain frente a terceros y hackers.

4 ¿Cómo funciona el Blockchain?

En el Capítulo 3, se han mostrado todos los elementos que forman una blockchain, en este capítulo 4, se estudia cómo interactúan estos elementos entre sí para crear un sistema que cumpla las características comentadas en el capítulo 2.

4.1 Transacciones

En el apartado 3.7.1, se ha estudiado que las transacciones son las operaciones **básicas** en el sistema Blockchain, en este apartado se indicará cómo son creadas.

4.1.1 Creación de la Transacción y firmado

Para crear una transacción en la Blockchain, es necesario rellenar los Inputs:

1. **Identificador de Transacción (TX ID)**
2. **Entrada (Origen):** dirección o direcciones públicas de origen que poseen los Tokens.
3. **Cantidad**: cantidad de Tokens que se enviarán en la transacción.
4. **Salida (Destino):** dirección o direcciones públicas de destino.

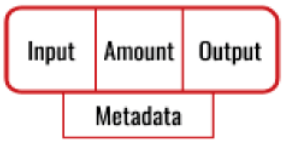


Figura 4.1 Transacción (RSK educate, 2015).

Esta tarea se realiza normalmente utilizando un Wallet.

El Wallet tiene conexión con la Blockchain donde comprueba el saldo disponible (la suma de las **UTXO**) en las direcciones de **Entrada** y, si existe suficiente, firma la transacción con la Clave **Privada** del usuario que quiere mandar la **cantidad** especificada de Tokens a la dirección de **Salida**, esta es la Dirección **Pública** del destinatario.

En definitiva se crea una salida de transacción con el estándar **P2PKH** (Pay-To-Public-Key-Hash) que contiene instrucciones que permite a cualquiera gastar esa salida si pueden probar que controlan la clave privada correspondiente a la clave pública. Estas instrucciones se conocen como el **scriptPubKey** (bitcoin.org, 2018).

4.1.2 Transmisión a la red

Una vez creada la transacción debe transmitirse a la Blockchain para que los nodos la recojan, validen e incorporen al Ledger.

El Wallet envía la transacción al nodo más **cercano** de la red. La transacción no necesita ser enviada inmediatamente después de su creación, es posible elegir cuando enviarla (RSK educate, 2015).

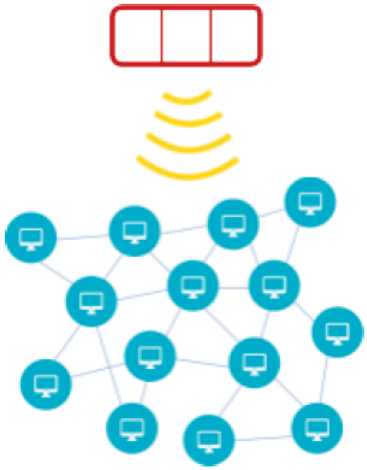


Figura 4.2 Broadcasting de una Transacción (RSK educate, 2015).

4.1.3 Propagación y verificación

Cuando la transacción llega al nodo más cercano, este la verifica y la **propaga** por la red y es **verificada** por el resto de nodos, ver figura 4.5, algunas verificaciones básicas son que existen suficientes tokens en la cartera de origen (UTXO previos conectados a esa dirección), estructura correcta, firma digital correcta, etc.

La firma digital se genera en función de la Clave Privada y la información de la transacción, si algún nodo malicioso **modifica** la transacción y la propaga, esta quedará **invalidada** debido a que la firma no podrá ser verificada, esa transacción será **rechazada** por los nodos.

- **Doble Gasto:**

  Puede darse la situación que un usuario cree dos transacciones **válidas** distintas, en una la transacción envía tokens a otro usuario como **pago** por un servicio o producto y en otra la transacción se los envía a sí mismo, los **UTXO** una vez gastados ya no pueden volver a utilizarse.

En teoría no existiría ningún problema, la primera transacción en registrarse sería la válida y la otra sería invalidada, salvo que las distintas transacciones lleguen a nodos distintos debido a los tiempos de propagación de la red, por tanto los nodos tendrían transacciones válidas **distintas** en sus Ledgers, y esto no debe ser posible.

En la figura 4.3, Alice genera 2 transacciones con los mismo UTXO, una a su beneficio y otra un pago a Bob, y las propaga por la red.

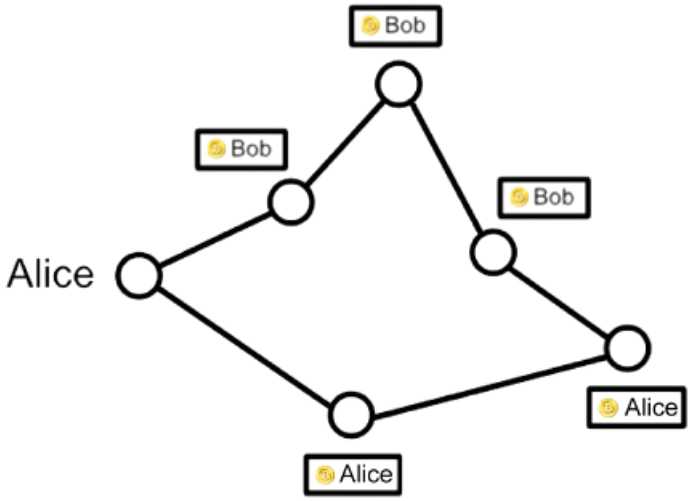


Figura 4.3 Alice propaga dos transacciones distintas (CuriousInventor, 2013).

Ambas transacciones son **válidas** porque su origen son UTXO **sin gastar**, para solucionar este problema aparece la cadena de bloques.

Los bloques encadenados marcan el **orden** de las transacciones, por tanto la primera transacción que se **añada** a un **bloque** será la transacción **válida** y la otra será **descartada**, por esta razón, no deben darse por válidas transacciones que no han sido confirmadas y añadidas a un bloque.

Existen en Blockchain 2 tipos de cadenas:

1. **Cadena de transacciones:** Historial de propiedad
2. **Cadena de bloques:** Orden de transacciones

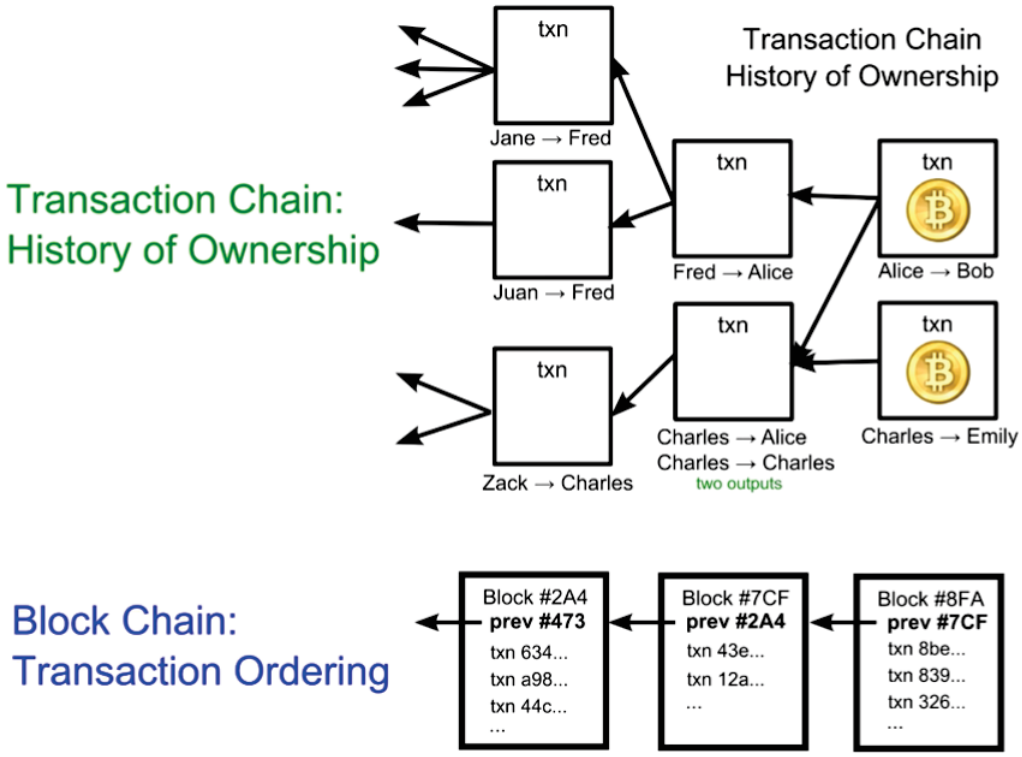


Figura 4.4 Tipos de cadenas en Blockchain (CuriousInventor, 2013).

Una vez que una transacción pasa la verificación, es enviada a la "**Mempool**" (abreviatura de Memory Pool) como transacción **No Confirmada** y espera hasta que un minero la seleccione para incluirla en el siguiente bloque.

El Mempool es el "**buffer**" de espera de cada nodo para las transacciones **pendientes**, a mayor velocidad se extraen de él las transacciones y se agregan a los bloques, mejor será la experiencia que obtendrán los usuarios.

Si la tasa de llegada de nuevas transacciones es más alta que la tasa de borrado/vaciado, se producirá un **"cuello de botella"** y las transacciones pueden tardar más **tiempo** en ser aprobadas (dependiendo de su tamaño y la comisión pagada al minero) (99bitcoins, 2019).

Cada nodo tiene una capacidad distinta para almacenar transacciones no confirmadas, con **BIP35** se permitió que los nodos pudieran acceder a las Mempool de otros nodos para permitir que:

- Nodos SPV pudieran conocer el estado de las transacciones antes incluso de ser añadidas a la Blockchain.
- Nodos mineros pudieran añadir más transacciones a su bloque o elegir aquellas con mayores comisiones y darles prioridad.
- Se realicen diagnósticos de red remotos.

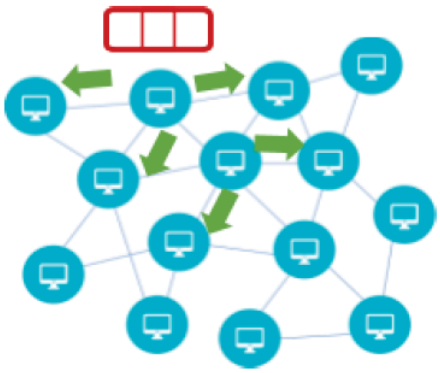


Figura 4.5 Propagación de una Transacción (RSK educate, 2015).

4.1.4 Validación

Una vez que la transacción está en el Mempool, los **mineros** recogen las transacciones (aquellas que pagan mayor comisión de transacción primero) y las **agrupan** en bloques.

En Bitcoin, a partir de mayo de 2017, cada bloque tiene un límite máximo de tamaño de 1 MB y permite recoger alrededor de 2.000 a 3000 transacciones, dependiendo del tamaño de cada transacción.

Una vez formado el bloque, siguiendo el algoritmo de **consenso** la red **valida** el bloque, y consecuentemente las **transacciones** incluidas en él.

Tras la validación del nuevo bloque, cada nodo lo añade a su Ledger, manteniendo todos los nodos un Ledger **común** e **inmutable**.

4.2 Mecanismos de Consenso principales

En este apartado se estudian cómo los algoritmos de consenso principales validan los bloques y transacciones de manera que se eviten ataques de nodos maliciosos.

4.2.1 Proof of Work (PoW)

Proof of Work o Prueba del Trabajo, es el mecanismo que utiliza medidas económicas para detener ataques y abusos en la red como los ataques de denegación de servicios (DDoS).

En el apartado 3.7.2, se indica que cada bloque tiene un encabezado con varios componentes: Versión, hash del bloque anterior, Raíz del árbol Merkle, Marca de tiempo, **Objetivo de dificultad**, **Nonce**, este último junto con el objetivo de dificultad, son los elementos más importantes en este mecanismo de consenso.

Minado

Una vez que el nodo minero ha rellenado el bloque con transacciones **no confirmadas**, el minero aplicará la función **SHA-256** al encabezado del bloque y obtendrá el **Hash del encabezado**, ver figura 4.6, este hash depende de valores **constantes** como son el Hash

del bloque anterior, la raíz Merkle, la marca de tiempo y el objetivo de dificultad, y de un valor **variable**, este valor es el **Nonce**.

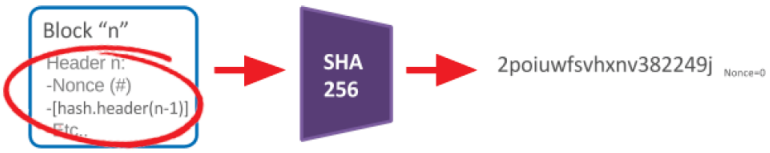


Figura 4.6 Minería Blockchain, búsqueda del Nonce (RSK educate, 2015).

Para saber si un Bloque es válido y pasa la prueba del trabajo, es decir se añade a la Blockchain y el minero recibe su recompensa, el **Hash** del encabezado generado debe cumplir el **objetivo de dificultad** marcado.

- **Objetivo de dificultad:**

  Es un parámetro Global de la Blockchain, varía para ajustar el **tiempo** de minería de cada bloque, de manera que si entra más capacidad de cálculo en la red (nodos mineros), la dificultad aumenta y si salen mineros, la dificultad disminuye para mantener el tiempo de generación de cada bloque **constante**.

  En Bitcoin la dificultad se cambia cada 2016 bloques basándose en el tiempo que le llevó generar los 2016 bloques anteriores. Si se encuentra un bloque cada 10 minutos, la búsqueda de 2016 bloques llevará exactamente 2 semanas. Si se encontraron los 2016 bloques anteriores en más de dos semanas, la dificultad se reducirá, y si se extrae de forma más rápida entonces ésta aumentará.

  El objetivo de dificultad es el mismo para todos los mineros de la red, y está definida por el **número de ceros** que debe contener a su comienzo el **Hash** del encabezado, siendo el valor máximo para este campo es 0x7fffffff, mientras que el mínimo es 0x008000 (bitcoinwiki.org, 2019a).

Un minero deberá utilizar la función SHA-256 sobre el encabezado del nuevo bloque **variando** el Nonce miles de veces hasta conseguir un Hash resultante que **cumpla** el objetivo de dificultad.

Validar el bloque por el resto de nodos de la red es una tarea **sencilla**, solo hay que realizar el SHA-256 del encabezado del nuevo bloque difundido por el minero y **comprobar** que cumple con la dificultad, pero obtener el Nonce es una tarea **ardua**, ya que requiere realizar la función SHA-256 (que es rápida y consume pocos recursos) **miles** de veces, hasta encontrar por **"fuerza bruta"** el Nonce correcto.

Una vez hallado el Nonce correcto, el siguiente paso es **transmitir** a la red el nuevo Bloque, cada vez que un nodo recibe un Bloque nuevo, lo **verifica** y lo **añade** a su Ledger.

Esta tarea es una carrera, el minero que encuentre la solución y la transmite a la red primero se lleva la **recompensa** del bloque.

En el momento que un minero que esté minando su bloque recibe un bloque nuevo desde

la red, significa que otro minero ya ha terminado el bloque antes que él y lo ha transmitido, por tanto **desecha** el bloque que estaba minando, y crea un **nuevo** bloque con transacciones no confirmadas, el **Hash del bloque anterior** (en este caso el bloque nuevo que le ha llegado) y el Nonce inicial.

En Bitcoin la dificultad ha aumentado tanto que la probabilidad de un nodo minero en solitario de encontrar el Nonce de un bloque es muy baja, por tanto se crearon los Pool de minería para aunar la potencia de varios nodos mineros.

La minería tiene 2 objetivos:

1. Verificar la legitimidad de una transacción y evitar el doble gasto.
2. Crear nuevas criptomonedas al recompensar a los mineros.

Problema de las ramas

Cuando **varios mineros**, minan un bloque prácticamente al mismo tiempo y lo transmiten a la red, los nodos añadirán aquél bloque que **reciban** primero, dando lugar a varías **ramas**, en el ejemplo son A, B y C, ver figura 4.7.

Esto es un proceso **natural** de la Blockchain debido a las latencias de la red, o causado por un intento de ataque de control de la Blockchain, donde la rama del atacante es finalmente desechada.

A partir de este momento, cada minero comienza a minar un **nuevo** bloque sobre su rama más **larga**, intentando ser el primero a nivel global. En el ejemplo es el Nodo 3 el que termina el Bloque D y lo propaga por la red, cuando los nodos reciban el Bloque D cambiarán **automáticamente** a la rama B, ya que esa es la rama más larga y comienzan a minar sobre el Bloque D.

En este caso se habrán creado tres ramas de la Blockchain **válidas** y **verdaderas**, como sólo puede existir un **Ledger** verdadero, el protocolo de consenso solventa este problema dando siempre prioridad la rama más larga, las otras ramas son **desechadas**.

En este caso una parte de los mineros habrán recibido el bloque A, otros el bloque B y otros el Bloque C, y empiezan a minar sobre esos bloques.

El primer minero que mine sobre el bloque recibido representará la cadena más larga cuando se transmita a la red.

En el ejemplo, se ha minado el bloque B partiendo de Hash del Bloque de B_{-1}, a continuación se ha minado el Bloque D. La Blockchain que contiene el bloque D será la Blockchain más larga, la verdadera y todos los nodos que tenían como Blockchain verdadera la de los Bloque A y C automáticamente la rechazan y adquieren la rama más larga (CuriousInventor, 2013).

Los Bloques A y C pasan a ser bloques huérfanos y los mineros que los minaron no reciben su recompensa y las transacciones que contenían vuelven a estar no confirmadas.

Tras desecharse un Bloque sus transacciones vuelven al Mempool, por tanto ya no están en la Blockchain, no son transacciones confirmadas.

En resumen, cuando se crea una transacción, esta pasa al Mempool donde es una transacción **no confirmada**, y puede ser **revertida**, una vez que es añadida a un bloque ya es

una transacción **confirmada**, pero aún puede ser revertida si se da el caso de múltiples ramas (en Bitcoin se da este caso alrededor de 1 vez al día), es muy poco probable que se den ramas de más de 2 bloques a la vez, por lo general, para grandes transacciones, se recomienda que existan **6 confirmaciones** o lo que es lo mismo, una rama con 6 bloques de longitud a partir de la transacción, lo que equivale a aproximadamente una hora de espera en Bitcoin, para dar por válida e **inmutable** la transacción.

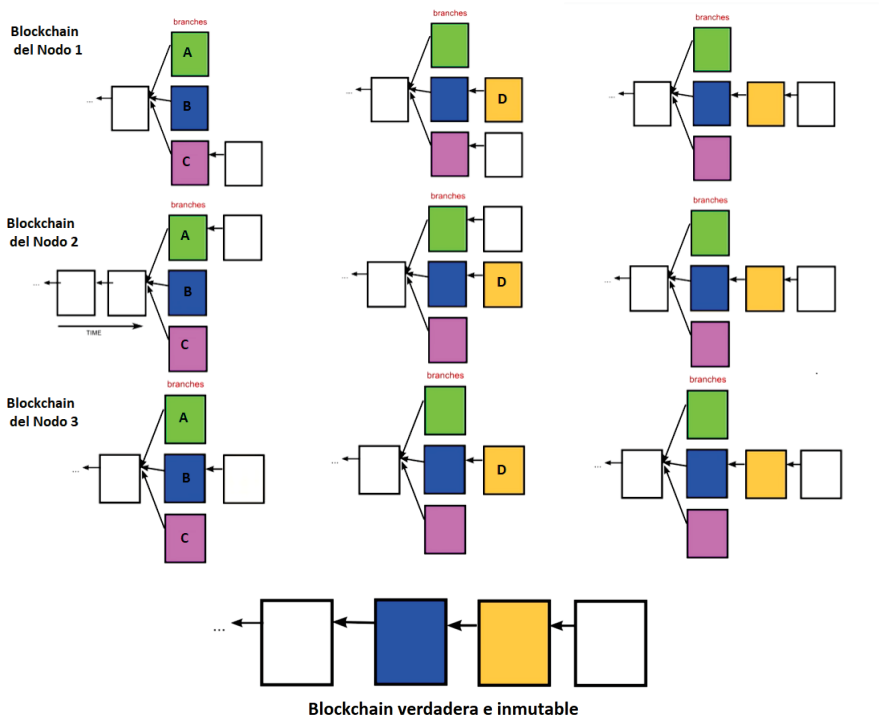


Figura 4.7 Minería Blockchain, rama más larga (CuriousInventor, 2013).

4.2.2 Proof of Stake (PoS)

Proof of Stake (PoS) es una categoría de algoritmos de consenso para Blockchains públicas que dependen de la participación económica de un validador en la red.

La prueba de participación permite que el mecanismo de consenso sea completamente **virtual**. Si bien el proceso general sigue siendo el mismo que la prueba de trabajo (PoW), el método para alcanzar la meta final es completamente diferente. En POW, los mineros resuelven enigmas criptográficamente difíciles utilizando sus recursos computacionales para validar bloques.

En las Blockchains públicas basadas en PoS, un conjunto de **validadores** se turnan para proponer y **votar** el siguiente bloque, y el peso del voto de cada validador depende del tamaño de su **depósito** (es decir, la participación). Las ventajas significativas de PoS incluyen **seguridad**, menor riesgo de centralización y **eficiencia** energética.

La cadena de bloques mantiene un registro de un conjunto de validadores, y cualquiera que tenga la criptomoneda base de la cadena de bloques puede convertirse en un validador al enviar un tipo especial de transacción que bloquea su criptomoneda en un depósito. El proceso de creación y aceptación de nuevos bloques se realiza a través de un algoritmo de consenso en el que pueden **participar** todos los validadores actuales (Buterin, 2019).

En un consenso distribuido basado en la prueba de trabajo, los mineros necesitan mucha energía. Y estos costes de energía se pagan con monedas fiduciarias, lo que lleva a una presión descendente constante sobre el valor de la moneda digital. En una investigación reciente, los expertos argumentaron que las transacciones de Bitcoin pueden consumir tanta electricidad como Dinamarca en 2020.

La comunidad de Ethereum quiere explotar el método de PoS para lograr una forma de consenso distribuida más **ecológica** y más **barata** (Rosic, 2017).

En la mayoría de los casos de prueba de participación, las unidades de moneda digital se crean en el lanzamiento de la moneda y su **número es fijo**. Por lo tanto, en lugar de usar unidades de criptomoneda como recompensa, los validadores reciben comisiones de transacción como recompensas. En algunos casos, se pueden crear nuevas unidades monetarias inflando el suministro de monedas, y los validadores pueden ser recompensados con nuevas unidades monetarias creadas, en lugar de comisiones de transacción (Ray, 2017).

¿Cómo se eligen a los validadores/forjadores?

Cuando se implemente **Casper** (el nuevo protocolo de PoS de Etherum), existirá un grupo de validadores/forjadores. Los usuarios pueden unirse a este grupo para ser seleccionados como el forjador. Este proceso estará disponible mediante la ejecución del Smart Contract de Casper al enviar Ether (la criptomoneda que alimenta la red Ethereum) (Rosic, 2017).

No habrá un límite impuesto en el número de validadores activos, pero se **regulará económicamente** reduciendo la tasa de interés si hay demasiados validadores y aumentando la recompensa si no hay suficientes.

Hay muchos tipos de algoritmos de consenso y muchas formas de asignar recompensas a los validadores que participan en el algoritmo de consenso.

Desde una perspectiva algorítmica, hay dos tipos principales:

1. En la prueba de la participación **basada en la cadena (chain-based PoS)**, el algoritmo **seudoaleatoriamente** selecciona **un validador** durante cada intervalo de tiempo (por ejemplo, cada período de 10 segundos), y asigna a ese validador el **derecho de crear** un solo bloque, y este bloque debe apunta a algún bloque anterior (normalmente el bloque al final de la cadena más larga anteriormente), y así, con el tiempo, la mayoría de los bloques convergen en una sola cadena en constante crecimiento.

2. En la PoS **estilo BFT (Byzantine Fault Tolerance)**, a los validadores se les asigna aleatoriamente el **derecho de proponer** bloques, pero acordar qué bloque es canónico se realiza mediante un proceso de múltiples rondas donde cada validador envía un "voto" para un bloque específico durante cada ronda, y al final del proceso, todos los validadores (honestos y en línea) **acuerdan permanentemente** si un bloque dado es parte de la cadena o no. Los bloques todavía pueden estar encadenados juntos; la diferencia es que el consenso sobre un bloque no depende de la longitud o el tamaño de la cadena posterior (Buterin, 2019).

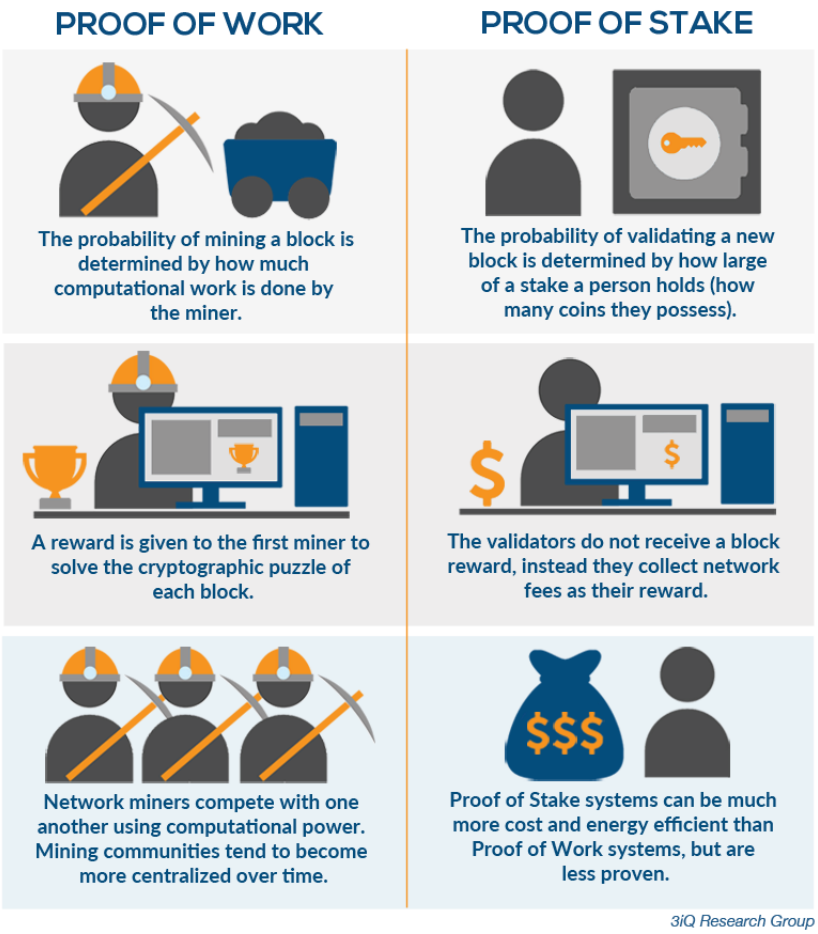


Figura 4.8 Diferencias PoW y PoS (Schumann, 2018).

PoW vs PoS

Algunos beneficios de PoS frente a PoW:

- No es necesario consumir grandes cantidades de electricidad para asegurar una cadena de bloques.
- Debido a la falta de alto consumo de electricidad, no es necesario emitir tantas monedas nuevas para motivar a los participantes a seguir participando en la red. En teoría, incluso puede ser posible tener una emisión neta negativa, donde una parte de las tarifas de transacción se quema/elimina y, por lo tanto, el suministro disminuye con el tiempo, generando una moneda deflacionaria y aumentando su valor.
- PoS permite una gama más amplia de técnicas que utilizan mecanismos de teoría de juegos para desincentivar la formación de cárteles centralizados y, si se forman, impide que utilicen métodos perjudiciales para la red.
- Reducción de los riesgos de centralización, ya que las economías de escala son un problema mucho menor. 10 millones de monedas le darán exactamente 10 veces mayores rendimientos que 1 millón de monedas, sin ninguna ganancia adicional desproporcionada porque en el nivel más alto pueda comprar mejores equipos de producción en masa, lo que es una ventaja para la Prueba de trabajo.

- La capacidad de aplicar sanciones económicas provoca que los ataques del 51 % son mucho más costosos de realizar que en la prueba de trabajo, parafraseando a Vlad Zamfir, "es como si su granja de ASIC se quemara si participó en un ataque del 51 %".

- Una red más segura a medida que los ataques se vuelven más costosos: si un hacker malicioso quisiera comprar el 51 % del número total de monedas para poder controlar la Blockchain, el mercado reacciona con una rápida apreciación de los precios.

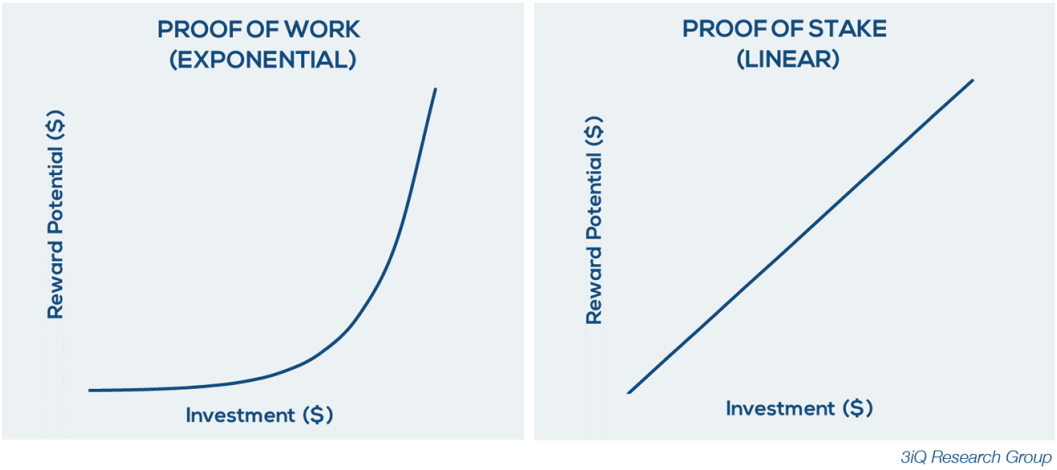


Figura 4.9 Economías de Escala (Schumann, 2018).

Casper será un protocolo de **fianzas** que se basa en un sistema de consenso económico. Los nodos (o los validadores) deben pagar una fianza/depósito para formar parte del consenso a través de la creación de nuevos bloques. El protocolo Casper determinará la cantidad específica de recompensas recibidas por los validadores gracias a su control sobre las fianzas.

Si un validador crea un bloque **"no válido"**, perderá su fianza, así como su privilegio de formar parte del consenso de la red.

Una vez que el forjador pone su participación, puede participar en el proceso de forja y, como ha apostado su propio dinero, en teoría ahora está **incentivado** para validar las transacciones correctas.

El sistema de seguridad de Casper se basa en algo parecido a las apuestas. En un sistema basado en PoS, las apuestas son las transacciones que, de acuerdo con las reglas de consenso, recompensará a su validador con un premio por cada bloque válido en la que el validador haya apostado.

Por lo tanto, Casper se basa en la idea de que los validadores apostarán de acuerdo con las apuestas de los demás y dejarán comentarios positivos que puedan acelerar el consenso (Rosic, 2017; Buterin, 2019; Ray, 2017).

4.2.3 Delegated Proof of Stake (DPoS)

La **Prueba de participación delegada** (también conocida como **DPoS**) es un algoritmo de consenso que mantiene un acuerdo irrefutable sobre la verdad en toda la red, valida

las transacciones y actúa como una forma de **democracia** digital.

La prueba delegada de participación utiliza la **votación** en tiempo real combinada con un sistema social de **reputación** para lograr el consenso. Es el protocolo de consenso **menos centralizado** en comparación con todos los demás, ya que es el más **inclusivo**. Cada titular de token puede ejercer un grado de influencia sobre lo que sucede en la red.

Los **delegados** activos son seleccionados por los poseedores de tokens por votación colocando sus tokens en nombre de su candidato (estos tokens no se gastan, solo representan la posición de la parte interesada y siguen siendo de su propiedad). El **poder de voto** que tiene el titular del token, también conocido como peso de voto, está determinado por la cantidad del token base que posee la cuenta.

Es importante que los delegados sean elegidos por el mejor interés de la red, ya que la mantienen funcionando sin problemas y de forma segura. En algunas versiones de DPoS, un delegado debe demostrar su compromiso depositando sus fondos en un depósito con bloqueo temporal (que se confisca en caso de comportamiento malicioso). Esta versión de DPoS a menudo se denomina prueba de participación basada en depósitos/fianzas.

Las funciones principales de los delegados están relacionadas con:

- Asegurarse de que su nodo está siempre en funcionamiento 24/7/365.
- Recopilación de las transacciones a través de la red en bloques.
- Firma y difusión de los bloques, validando las transacciones.
- Gobernar el sistema y proponer cambios centrales.
- Si hay problemas con respecto al consenso, DPoS permite resolverlos de manera justa y democrática.

Los delegados no tienen el poder de cambiar los **detalles** de la transacción, pero como son validadores, teóricamente podrían excluir ciertas transacciones en un bloque. Sin embargo, esto tiene muy poco efecto ya que el siguiente bloque creado incluirá estas transacciones, dando al próximo delegado las tarifas asociadas con la validación de las mismas. Como tal, las transacciones sólo se retrasarán ligeramente.

Además, esto inevitablemente conduciría a que el delegado deshonesto sea **expulsado** por el resto de la red. En esencia, una red DPoS es **autogestionada** y **vigilada** por todos sus participantes, lo que garantiza que los intereses de la red sigan siendo la prioridad.

Algunos blockchains con DPoS ofrecen la oportunidad de **reelegir** a los delegados si el resultado de sus decisiones no es aprobado por sus votantes. La votación es un proceso **continuo**, por lo que cada delegado está bajo la presión de perder su lugar frente a otro competidor más popular.

Las pérdidas de reputación y financieras son una motivación primordial para desincentivar comportamientos maliciosos.

La mayoría de las cadenas de bloques basadas en DPoS tienen en cuenta la cantidad de

tokens en propiedad, pero no existe una regulación que impida a los usuarios votar debido a que su participación no es lo suficientemente grande. El hecho de que la oportunidad de votar se otorgue a cada usuario de la red es lo que hace que DPoS sea el enfoque más democrático para el algoritmo de consenso de cadena de bloques.

Las cadenas de bloques de PoW no permiten que los usuarios con poca potencia de computación influyan realmente en la red, esa es la razón principal de la existencia de pools de minería, que actualmente son las únicas entidades que rigen Bitcoin.

La mayoría de las cadenas de bloques de PoS excluyen a los usuarios con pequeñas participaciones de tomar decisiones sobre el gobierno de la red.

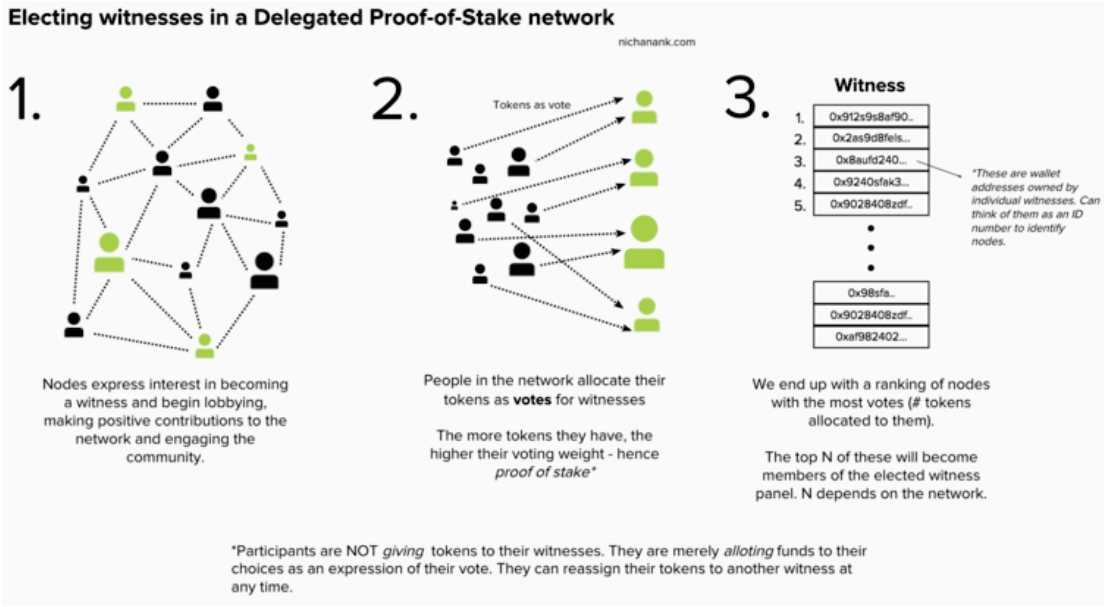


Figura 4.10 Algoritmo de Consenso DPoS (Winterfield, 2018).

Ventajas y desventajas de la DPoS

DPoS no solo es un sistema más **democrático** (da voto a los pequeños usuarios), sino que también es más **eficiente** y **efectivo**. La selección de creadores de bloques permite **validar** las transacciones en cuestión de segundos, más **rápido** que PoW y PoS.

Por ejemplo, en el caso de Lisk, actualmente solo tarda 10 segundos validar un bloque de transacciones.

Los delegados reciben comisiones de transacción como recompensa por ejecutar los nodos que procesan y validan las transacciones que se realizan a través de la red, así como recompensas mensuales por el mantenimiento de la red que, con el tiempo, se reduce gradualmente. Solo puede haber un cierto número de delegados a la vez y estos están determinados por un **sistema electoral competitivo**, en el que todos y cada uno de los usuarios pueden emitir voto según su preferencia.

La supervivencia de la red requiere la participación y coordinación, mediante la votación para eliminar e introducir delegados, de una comunidad interesada en un gobierno efectivo.

Los sistemas DPoS son vulnerables a la centralización, ya que el número de delegados está estrictamente **limitado**.

La cadena de bloques DPoS está expuesta a los fallos de la votación en la vida real clásica.

Por ejemplo, los usuarios de DPoS con **poco poder** de voto pueden decidir que su voto no importa en comparación con los votos de las partes interesadas más grandes (Academy, 2019; Bitcoin.org, 2019).

4.2.4 Stellar Consensus Protocol (SCP)

El Stellar Consensus Protocol (SCP), es un sistema de **Acuerdo Bizantino Federado (FBA)** que permite a las redes informáticas **descentralizadas** y sin **líder** alcanzar de manera eficiente un resultado de **consenso** en alguna decisión. La Blockchain Stellar utiliza **SCP** para proporcionar una visión coherente del historial de transacciones de la red a todos los participantes.

Cualquier sistema de consenso en una red de computación distribuida debe ser tolerante a errores: debe producir resultados consistentes a pesar de errores como problemas de comunicación, nodos que no responden y mensajes mal ordenados.

Un sistema de **Acuerdo Bizantino** además tolera los fallos "bizantinos": nodos que proporcionan información falsa, ya sea debido a un error o en un intento deliberado de subvertir el sistema u obtener alguna ventaja.

este sistema evita fallos bizantinos usando una regla de mayoría llamada **quórum**. Un nodo en dicha red se niega a comprometerse con una versión particular de la historia hasta que ve que una cantidad suficiente de sus nodos vecinos también están preparados para comprometerse, un quórum. Una vez que eso sucede, han formado un **bloque de votación** lo suficientemente grande como para obligar a los nodos restantes en la red a estar de acuerdo con su decisión.

Para formar un quórum se necesita al menos una mayoría y más típicamente una mayoría mayoritaria para combatir errores y fraudes, saber cuándo tiene la mayoría significa saber cuántos participantes tiene en total. Pero si su colección de participantes es una red poco definida a la que los miembros pueden **unirse y salir** a voluntad, sin necesidad de coordinar con ninguna **autoridad central**, entonces es necesario un sistema de **Acuerdo Bizantino Federado**: uno que pueda determinar los quórumes no a partir de una lista predeterminada de nodos, sino dinámicamente, a partir de una lista siempre **cambiante** e inevitablemente **incompleta** de los nodos en un momento determinado (Glickstein, 2019).

Acuerdo Bizantino Federado

Para llegar a Quórum en el protocolo SCP:

1. Los nodos llevan a cabo rondas de votación federada en "nominaciones". Una ronda de votación federada significa:
    - Un nodo emite un voto para alguna declaración, como "Nomino el valor V"
    - El nodo escucha las respuestas de sus vecinos hasta que encuentra uno que comunica "aceptar"
    - El nodo busca un "**quórum slice**" (una pequeña parte del quórum total que será necesario alcanzar) con los nodos que también aceptan la declaración. Esto "confirma" la declaración.

2. Tan pronto como un nodo puede confirmar uno o más candidatos, comienza a tratar de "preparar" una "propuesta" a través de más rondas de votación federada.

3. Tan pronto como un nodo puede verificar que una propuesta está preparada, comienza a tratar de "consolidar" la propuesta a través de aún más rondas de votación federada.

4. Una vez que un nodo puede confirmar que se ha consolidado una propuesta, puede "externalizar" el valor de esa propuesta, usándola como resultado del consenso.

Estos pasos implican varias rondas de votación federada que colectivamente forman una sola ronda de SCP, véase la figura 4.11 como ejemplo.

En un sistema **FBA**, los nodos se transmiten entre sí en qué porciones de quórum **confían**. Los nodos determinan **individualmente** en qué quórum confiar y construyen su propio quórum en función de la información que tienen a mano. La situación ideal es tener **intersecciones de quórum** o superposiciones de quórum. Esto asegura que haya **consenso** en toda la red. Si no hay intersecciones, habrá quórumes inconexos que pueden llevar al registro de transacciones contradictorias.

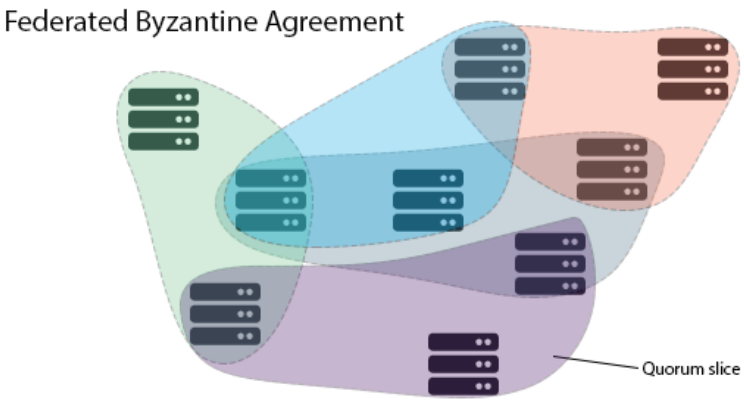


Figura 4.11 Acuerdo Bizantino Federado (FBA) (Renders, 2018).

Ventajas del Stellar Consensus Protocol

Blockchain usa la tolerancia a fallos bizantina (**BFT**) para mantener la confiabilidad de los registros y transacciones. La mayoría de las cadenas de bloques utilizan lo que se conoce como Tolerancia de falla bizantina práctica (**PBFT**). Una red de blockchain que usa PBFT necesita el 66 por ciento de los validadores para acordar transacciones para lograr quórum o consenso. Para mantener la validez de la red, el número de nodos maliciosos debe permanecer por debajo del 33 por ciento.

La PBFT no es lo suficientemente tolerante a fallos. Además, la selección de validadores hace que los sistemas PBFT se vuelvan **centralizados**.

En un sistema **FBA**, no hay necesidad de validadores predefinidos. Los nodos toman las decisiones **individualmente** sobre en quién confiar. Se **descentraliza** el sistema. En lugar de todo el quórum o un número preestablecido de nodos, los sistemas FBA utilizan el "**quórum slice**" o un subconjunto del quórum.

4.3 Tecnología Blockchain - Observaciones

Llegados a este punto, ya es posible visualizar las ventajas de la tecnología Blockchain y el futuro recorrido de esta tecnología.

El por qué es importante y dónde reside su valor, debido a sus propiedades de seguridad, inmutabilidad y confianza.

En los capítulos 3 y 4 se ha estudiado qué es la cadena de bloques y cómo funciona para permitir crear un sistema de consenso confiable y distribuido, esto implica que si se desea enviar y/o recibir transacciones con alguien no se necesita confiar en servicios de terceros.

En el capítulo 5 se va a estudiar cómo se interactúa con la Blockchain hoy en día y en el capítulo 6 cómo podría integrarse esta tecnología con la disciplina del Control Automático.

5 Criptomonedas, Smart Contracts, Dapps e Industria 4.0

En el presente capítulo se estudia la evolución de la tecnología Blockchain, se pueden identificar 4 etapas/generaciones (Darko, 2019):

- **Blockchain 1.0: Moneda**

  Las Blockchains de 1ª generación son Blockchains de consenso por Prueba de Trabajo, donde los nodos con mayor potencia computacional, tienen el mayor poder de voto y donde el único propósito de la Blockchain es ser un **procesador de pagos**.

- **Blockchain 2.0: Contratos inteligentes**

  Las Blockchains de 2ª generación son cadenas de bloques de Prueba de Trabajo que tienen más funciones que simplemente ser un procesador de pagos. Se trata de Blockchains, como Ethereum o Neo, que permiten ejecutar c**ontratos inteligentes**, lanzar ICOs y ejecutar DApps en su plataforma.

- **Blockchain 3.0: DApps**

  El problema de las cadenas de bloques de 1ª y 2ª generación es que no son escalables en absoluto, tardan mucho tiempo en confirmar las transacciones, están muy centralizadas a través de grupos de minería, tienen comisiones de transacción y consumen mucha energía.
  Las Blockchains de 3ª generación son Blockchains que ya ni siquiera son Blockchains en un sentido práctico, lo que les permite ir más allá de las restricciones mencionadas anteriormente que trae la arquitectura común de Blockchain, por ejemplo, IOTA con su Tangle tiene una estructura de **Grafos Acíclicos Dirigido** (DAG), ver apartado 3.3.2.
  Las Blockchains de 3ª generación son arquitecturas de software descentralizadas que permiten una **escalabilidad** casi infinita, tienen transacciones **instantáneas**, **descentralización** casi infinita, no cobran **comisiones** y solo utilizan una millonésima parte de la **energía** de la que utiliza Bitcoin.

- **Blockchain 4.0: Integración en la industria 4.0**

  Los proyectos de Blockchain 3.0 fallan en ofrecer una plataforma descentralizada y abierta que abarque **negocios del mundo real** de cualquier tipo, tamaño o volumen.

La industria de la cadena de bloques está ansiosa por una red de alto rendimiento y de acceso libre que brinde infraestructura suficiente para que los desarrolladores de DApps reduzcan los costos de desarrollo y reduzcan el ciclo de desarrollo.
La llegada de Blockchain 4.0 marcará un hito destacado en la historia de la evolución de Blockchain. Blockchain 4.0 será la cadena de bloques de alto rendimiento y "amigable" para los negocios que pueda ejecutar **casos de uso y aplicaciones del mundo real**.

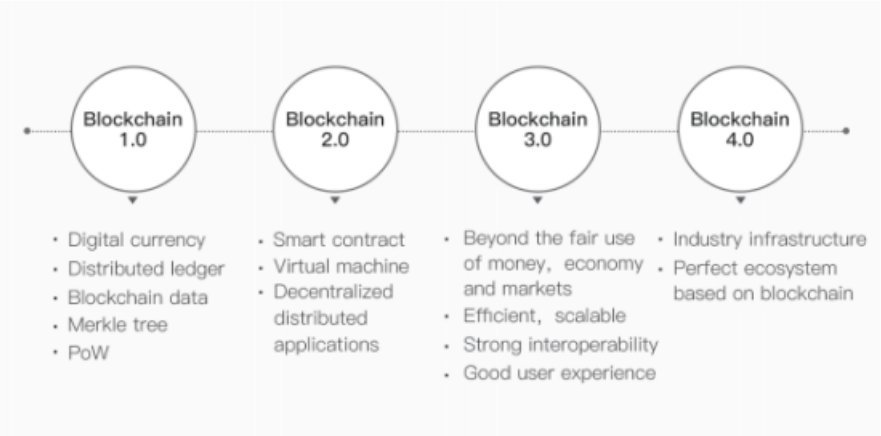


Figura 5.1 Evolución de Blockchain (InterValue, 2018).

5.1 Blockchain 1.0: Criptomoneda

La implementación de la tecnología de "libro de cuentas" distribuido condujo a su primera aplicación: las criptomonedas.

Una criptomoneda, es un medio **digital** de intercambio que utiliza criptografía para asegurar las **transacciones** financieras, controlar la creación de unidades adicionales y verificar la transferencia de activos.

Las criptomonedas son un tipo de divisa alternativa y de moneda digital, tienen un control **descentralizado**, en contraposición a las monedas centralizadas y a los bancos centrales. Esto permite que las transacciones financieras utilizando la tecnología Blockchain se ejecuten sin depender de **servicios de terceros**, siendo Bitcoin el ejemplo más destacado en este segmento. Se diseñó como "efectivo para Internet", un sistema de pago digital con objetivo de desarrollar el "Internet del dinero/valor" (En Wikipedia, s.f.[k]; InterValue, 2018; Unibright.io, 2017).

5.2 Blockchain 2.0: Contratos Inteligentes

Los **Smart Contracts o Contratos Inteligentes**, son pequeños programas de software alojados en la cadena de bloques. Son programas informáticos **autónomos** que se ejecutan **automáticamente** y las condiciones se definen de antemano, como la facilitación, la verificación o la ejecución del cumplimiento de un contrato.

Una gran ventaja que ofrece esta tecnología, es la cadena de bloques que hace **imposible manipular** o **piratear** contratos inteligentes. Por lo tanto, los contratos inteligentes reducen el costo de la verificación, la ejecución, el arbitraje y la prevención del fraude y

permiten una definición **transparente** del contrato para superar el problema del riesgo moral.

El Blockchain más destacado en este campo es el Blockchain de Ethereum (InterValue, 2018; Unibright.io, 2017).

5.2.1 ¿Qué es un smart contract?

Un contrato inteligente es un "contrato" que es capaz de ejecutarse y hacerse cumplir por sí mismo, de manera autónoma y automática, sin intermediarios ni mediadores.

Mientras que un contrato estándar describe los términos de una relación (generalmente uno exigible por ley), los smart contracts evitan el lastre de la interpretación de los contratos estándar al no ser verbal o estar escrito en los lenguajes comunes. Los smart contracts se tratan de códigos informáticos escritos con **lenguajes de programación**, siendo los términos del contrato puras sentencias y comandos en el código que lo forma.

Un smart contract puede ser creado y llamado por personas **físicas** y/o **jurídicas**, pero también por **máquinas** u otros **programas** que funcionan de manera autónoma.

Un smart contract tiene validez, sin depender de autoridades, debido a su naturaleza: es un código visible por todos y que no se puede cambiar al existir sobre la tecnología Blockchain, la cual le da ese carácter descentralizado, inmutable y transparente.

Los Smart Contracts son programas en la nube que siempre actúan **igual**, y permiten almacenar información que no puede ser modificada a traición.

Son los programas más seguros jamás creados en la humanidad y solo fallan cuando están **mal programados**.

Los Smart Contracts son capaces de gestionar **activos** digitales, sujetos a un determinado valor económico, por lo que los Smart Contracts pueden gestionar dinero. Esto requiere que se haga especial énfasis en la correcta programación del mismo, pues el Smart Contract podría tener fallos de seguridad o fallos que generasen errores de ejecución o comportamientos inesperados.

Para que todo esto sea posible, tiene que haber un proceso completamente seguro que garantice que, al menos dos partes, puedan ejecutar el contrato sin necesidad de confiar el uno del otro, ni tan si quiera conocerse (Bit2Me, 2018b).

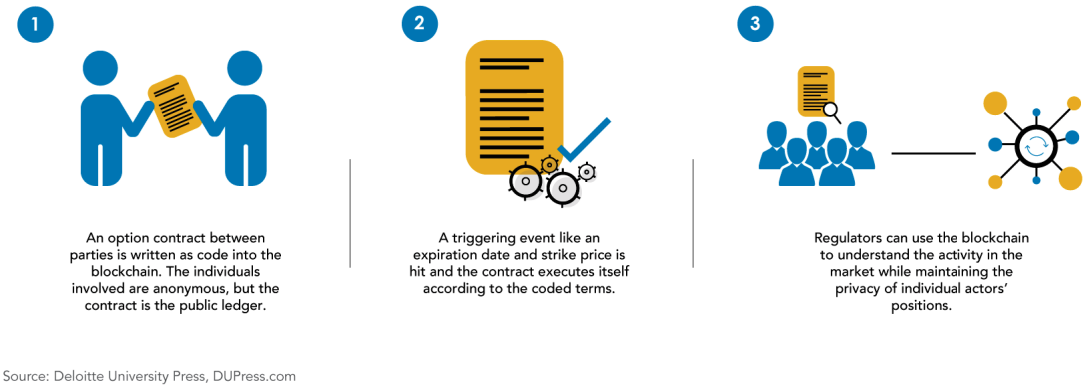


Figura 5.2 Contratos en la Blockchain (Kristen Silverberg, 2016).

5.2.2 La Blockchain de Etherum

Ethereum, es uno de los proyectos más conocidos en el sector de los smart contracts, es una plataforma de **computación distribuida** basada en una blockchain pública y que además permite ejecutar contratos inteligentes P2P (entre los nodos, sin servidores centrales) en una máquina virtual **descentralizada** llamada **Ethereum Virtual Machine (EVM)**.

Etherum posee todas las características de la tecnología Blockchain, es distribuido, tiene su propia criptomoneda, mineros y bloques, la diferencia es que ha creado un intérprete de lenguaje de programación mucho más extenso (**Turing completo**), permitiendo añadir lógica mucho más compleja dentro de la Blockchain, admite un conjunto más amplio de instrucciones computacionales.

Reemplaza el lenguaje más restrictivo de Bitcoin (un lenguaje de secuencias de comandos de alrededor de cien scripts) y lo reemplaza con un lenguaje que permite a los desarrolladores escribir sus propios programas.

Es similar a un ordenador **distribuido**, el cual utiliza su criptomoneda (Ether) como el pago proporcional a la potencia computacional requerida para que los mineros ejecuten el programa/contrato inteligente. Es decir, ahora con Ethereum los Smart Contracts son programas con muchas más funcionalidades y posibilidades (Bit2Me, 2018b; Hertig, 2017a).

Estructura

La estructura de la cadena de bloques de Ethereum es muy similar a la de Bitcoin, ya que es un registro compartido de todo el historial de transacciones. Cada nodo en la red almacena una copia de este historial.

La gran diferencia es que sus nodos almacenan el estado más **reciente** de cada contrato inteligente, además de todas las **transacciones** de Ether.

Para cada smart contract, la red debe realizar un seguimiento del "estado" o la información actual de todas estas aplicaciones, incluido el saldo de cada usuario, todo el código de contrato inteligente y dónde está almacenado.

Bitcoin usa resultados de transacciones no gastados **UTXO** para rastrear el saldo de bitcoin y para realizar transacciones futuras, la red de Bitcoin debe sumar todos sus cambios, que se clasifican como gastados o no gastados.

Ethereum, por otro lado, utiliza las **cuentas**.

Al igual que los fondos de cuentas bancarias, las fichas de Ether aparecen en una wallet y se pueden transferir a otra cuenta. Los fondos siempre están en alguna parte, pero no tienen lo que podríamos llamar una relación continua, la figura 5.3 muestra estas diferencias.

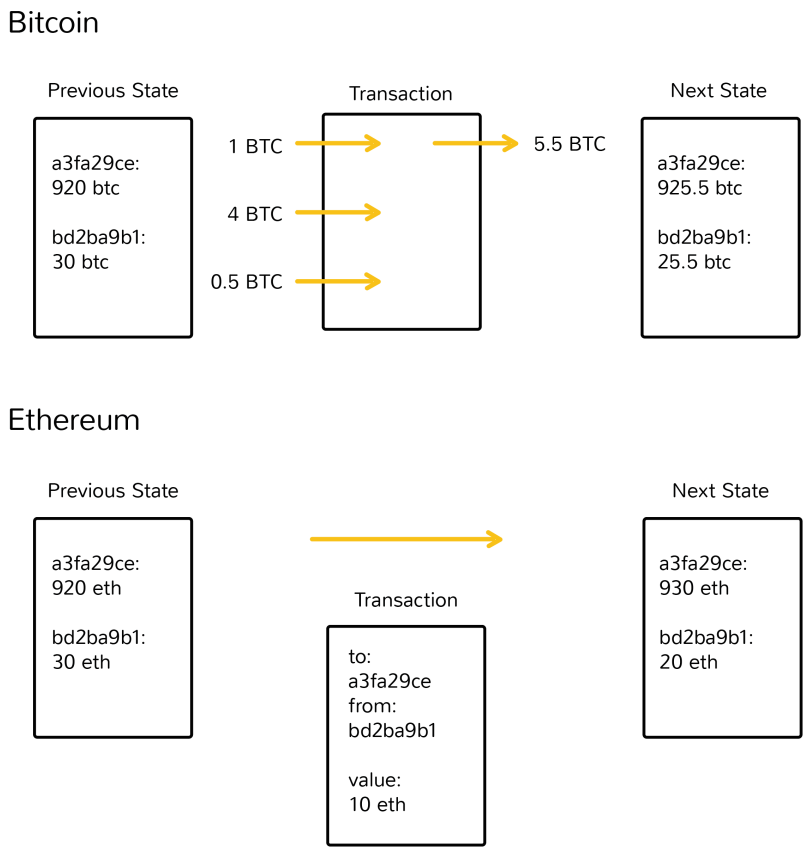


Figura 5.3 Estructura Bitcoin vs Etherum (Hertig, 2017b).

Etherum Virtual Machine (EVM)

Con Ethereum, cada vez que se ejecuta un programa, una red de miles de nodos lo procesa.

Los programas escritos en un lenguaje de programación específico para el contrato inteligente se compilan en "**bytecode**", que permite a la máquina virtual de Ethereum leer y ejecutar.

Todos los nodos ejecutan este contrato utilizando sus EVMs.

Cada nodo en la red contiene una **copia** de la transacción y el historial de contratos inteligentes de la Blockchain, además de mantener un registro del "estado" actual. Cada vez que un usuario realiza alguna acción, todos los nodos de la red se ponen de acuerdo en que este cambio tuvo lugar mediante mecanismos de consenso.

El objetivo aquí es que la **red** de mineros y nodos asuma la **responsabilidad** de transferir el cambio de estado a estado, en lugar de alguna autoridad externa.

Los mineros de Bitcoin validan el cambio de propiedad de los bitcoins de una persona a otra. El EVM ejecuta un contrato con las reglas que el desarrollador programó inicialmente.

El cálculo real en el EVM se logra a través del lenguaje bytecode de bajo nivel, pero los desarrolladores pueden escribir contratos inteligentes en lenguajes de alto nivel como **Solidity** y **Serpent** que son más fáciles de leer y escribir por las personas.

Los mineros son los que evitan el comportamiento **dañino**, como asegurarse de que nadie esté realizando doble gasto y rechazar los contratos inteligentes por los que no se ha pagado.

Hay unos cuantos miles de nodos Etherum, y cada nodo está compilando y ejecutando el **mismo código** (Hertig, 2017b).

5.2.3 Los oráculos

Una de las características más importantes para que un smart contract pueda **interactuar** con el mundo **real**, son los llamados **oráculos** (oracles en inglés).

Los oráculos son instrumentos que permiten actualizar estados internos de un smart contract a través de información del exterior (generalmente obtenida a través de APIs).

Los oráculos también funcionan de forma **autónoma**. No obstante, hay que tener presente que la fuente que usa el oráculo se trata de una **tercera parte** en la que hay que confiar, y que podría corromperse, o simplemente podría fallar su servidor, algo que tiene implicaciones negativas: estamos centralizando la confianza.

Ya existen proyectos que están desarrollando soluciones para este problema combinando el resultado de todos los proveedores de información que se le indique y es este quien determina su decisión en función de lo que la mayoría le diga (Bit2Me, 2018b).

5.3 Blockchain 3.0: DApps

DApp es una abreviación de **Aplicación Descentralizada**, una aplicación que evita la infraestructura centralizada.

Utiliza el almacenamiento descentralizado y la comunicación descentralizada, por lo que la mayoría de las DApps tienen su código de fondo ejecutándose en una **red descentralizada** de igual a igual (P2P), una cadena de bloques.

En contraste, una aplicación tradicional tiene su código backend ejecutándose en servidores **centralizados**. Un DApp puede tener un código de frontend e interfaces de usuario escritas en cualquier lenguaje que pueda hacer llamadas a su backend, como una aplicación tradicional, o puede tener su interfaz alojada en almacenamientos descentralizados como Ethereums Swarm (Unibright.io, 2017).

$$DApp = Frontend + Smart\,Contracts$$

El frontend de una aplicación descentralizada representa lo que se ve, y el backend representa toda la lógica de negocio (Business).

Esta lógica de negocio está representada por uno o varios **contratos inteligentes** que interactúan con la cadena de bloques subyacente.

La interfaz, así como archivos, fotos, vídeos o audio, se pueden alojar en redes de almacenamiento descentralizado como Swarm o IPFS.

La diferencia con las aplicaciones web tradicionales es que utilizan HTML, CSS y javascript o similares para representar una página web. Esta página interactúa con una **base de datos centralizada**, donde se almacenan todos los datos. Cuando utiliza un servicio como Twitter, Facebook, Amazon o Airbnb, por ejemplo, la página web llamará a una API para procesar sus datos personales y otra información necesaria almacenada en sus servidores, para mostrarlos en la página.

La identificación de usuario y las contraseñas se utilizan para la identificación y la autenticación, con **bajos niveles de seguridad**, ya que los datos personalizados se almacenan en el servidor del proveedor de servicios.

$$Webs\,Tradicionales : FrontEnd \rightarrow API \rightarrow Database$$

Las aplicaciones descentralizadas son similares a una aplicación web tradicional. La interfaz utiliza exactamente la misma tecnología para renderizar la página.

Contiene una **wallet** que se comunica con el blockchain. La wallet gestiona las claves criptográficas y la dirección de Blockchain. La infraestructura de **clave pública** se utiliza para la identificación y autenticación del usuario. En lugar de que una API se conecte a una base de datos, un software de wallet activa las actividades de un **contrato inteligente**, que interactúa con una cadena de bloques.

$$DApp : Front\,End\,(incluye\,wallet) \rightarrow SmartContract \rightarrow Blockchain$$

A diferencia de las aplicaciones $Web\,2$, las aplicaciones $Web\,3$ necesitan una conexión a la cadena de bloques, que es administrada por una wallet que mantiene un registro de las claves privadas y la dirección de la cadena de bloques, que representa la identidad y el punto de referencia únicos.

Sin un software que gestione nuestra identidad digital, no podremos interactuar con la cadena de bloques. El $Web\,3$, por lo tanto, se basa en la tecnología actual de $Web\,2$ e introduce elementos adicionales en la **capa de aplicación**.

En el backend, $Web\,3$ agrega una nueva **capa de infraestructura** para que las aplicaciones descentralizadas interactúen con los protocolos **descentralizados**. Las aplicaciones descentralizadas deben tener un componente que administre las claves privadas de un usuario, con las cuales se pueden firmar transacciones en la **capa de estado**, el Blockchain (blockchainhub, 2019).

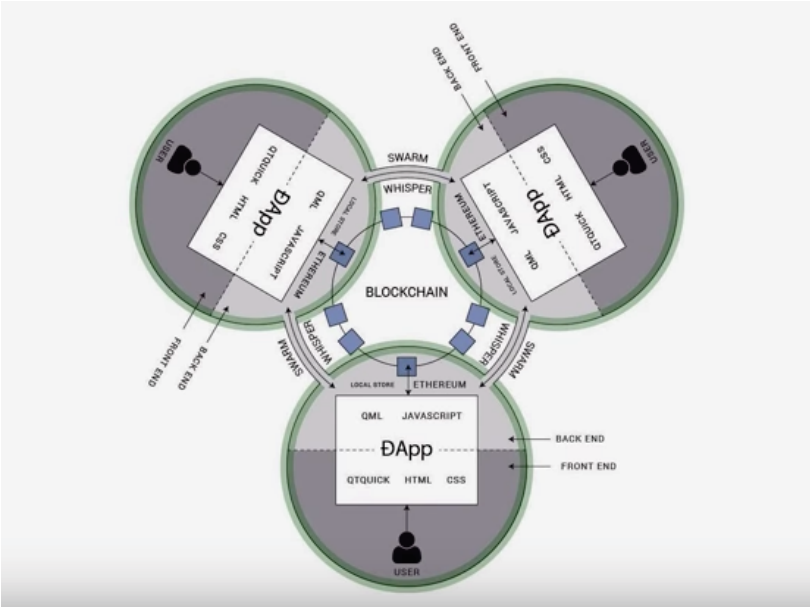


Figura 5.4 Estructura DApp (blockchainhub, 2019).

5.4 Blockchain 4.0: Haciendo blockchain utilizable en la industria 4.0

Blockchain 4.0 describe soluciones y enfoques que hacen que la tecnología de Blockchain sea utilizable para las demandas comerciales. Especialmente las exigencias de la industria 4.0.

La industria 4.0 significa, a corto plazo, automatización, planificación de recursos empresariales e integración de diferentes sistemas de ejecución.

Sin embargo, esta revolución industrial exige un mayor grado de **confianza** y protección de la **privacidad**: aquí es donde entra en acción la cadena de bloques.

Al agregar blockchain a los sistemas de TI se logra la **integración** en el negocio, lo que permite procesos cruzados entre Blockchain y los sistemas de producción, por ejemplo, que las máquinas realicen un pedido de forma segura y autónoma para que lleguen sus piezas de repuesto.

La gestión de la cadena de suministro, las transacciones financieras y los pagos basados en condiciones, la recopilación de datos de IoT, la gestión de la salud y la gestión de activos son solo algunos ejemplos de las áreas que pueden ser mejoradas por la tecnología Blockchain.

Mientras que Blockchain 3.0 busca cubrir las demandas de consumo de la vida real, Blockchain 4.0 busca satisfacer las demandas de la Industria 4.0 (Unibright.io, 2017).

Blockchain 4.0 permitirá a las empresas **trasladar** algunas o todas sus actividades actuales a aplicaciones seguras basadas en ledgers descentralizados, confiables y encriptados. Pone por primera vez las **ventajas** básicas e inherentes de Blockchain al **alcance** de las empresas.

Las tareas administrativas de las empresas, pueden ser automatizadas, pero en la cadena de bloques, están automatizadas como algo habitual,permiten una integración inme-

diata. El peligro de pérdida de datos se elimina casi por completo. Y un registro inmutable significa que los datos son precisos, lo que lleva a una mejor previsión y planificación.

Estas ventajas están integradas en toda la tecnología de Blockchain, pero sin el conjunto de características "4.0", las empresas no pueden acceder a ellas.

Existen beneficios empresariales adicionales únicos para los ecosistemas de Blockchain 4.0. Un Marco de Desarrollo Integrado (IDE) ofrece a las empresas la oportunidad de construir sus propios Dapps, los esfuerzos de compatibilidad entre Blockchains distintas significan que la mayoría de los Dapps se ejecutarán en múltiples Blockchains de nivel 4.0 (Official, 2018).

6 Blockchain en la Automática

6.1 Control Automático

6.1.1 Introducción

El **Control Automático** es una rama de la ingeniería que se ocupa del control de un proceso en un estado determinado.

El Control Automático estudia el comportamiento de los sistemas **dinámicos** y la **actuación** necesaria sobre este para controlarlo.

En general, la entrada al sistema es una señal analógica o digital y la salida del sistema corresponde a la variable que se pretende controlar.

Cuando una o más de las variables de salida de un sistema tienen que seguir el valor de una referencia que cambia con el tiempo, se necesita interponer un **controlador** que manipule los valores de las señales de entrada al sistema hasta obtener el **valor deseado** de salida (En Wikipedia, s.f.[t]).

6.1.2 Sistemas de Control

Podemos definir un **sistema de control** como la combinación de elementos que, **actuando** sobre una planta o proceso, trata de fijar alguno de sus parámetros o de hacer que varíe, en el transcurso del tiempo, de una forma determinada que se predefine.

Para representar esquemáticamente un sistema se utilizan los diagramas de bloques, en los que cada elemento o conjunto de elementos se simboliza con un bloque. Unas flechas indican el sentido de la información, que es único (En Wikipedia, s.f.[t]).

El control automático siempre está integrado por tres operaciones básicas:

- **Medida:** la medida de la variable a controlar se realiza mediante un sensor.
- **Decisión:** basándose en la medida, el controlador debe **decidir** qué hacer, para mantener la variable en el valor deseado.
- **Acción:** como resultado de la decisión del controlador, el sistema debe emprender alguna acción. Normalmente la realiza el elemento final u órgano de control.

Lazos de Control

- **Lazo Abierto:**

  En el control de **lazo abierto**, el controlador **recibe** una señal que puede ser externa al sistema y, en función de ella, **envía** una orden al órgano de control, pero no vuelve a recibir **información** sobre lo que ocurre en el proceso una vez ejecutada la acción.

  Son sistemas de control en los que el valor actual de la variable no tiene efectos sobre la acción de control. El control deberá, una vez recibida una entrada de referencia, suministrar a la planta o proceso una salida adecuada, para que la magnitud a controlar evolucione de la forma prevista.

  No se mide la variable, ni se realimenta para compararla con la señal de entrada. Cuando un control está en manual el lazo es abierto.

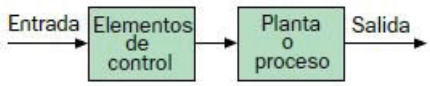


Figura 6.1 Control Lazo Abierto (Mateo Cruz, 2019).

- **Lazo Cerrado:**

  Para evitar los problemas del control en lazo abierto, la teoría de control introduce la **realimentación**.

  El regulador de lazo cerrado utiliza la realimentación para controlar los estados y las salidas de un sistema dinámico. El nombre de "lazo cerrado" hace referencia al camino que sigue la información en el sistema: la entrada al proceso afecta a la salida del mismo . La **salida** se mide con sensor y una vez **comparada** con la **referencia** o consigna, se procesa mediante el controlador o regulador; el resultado, una señal de control, se reenvía a la entrada del proceso, cerrando el lazo (En Wikipedia, s.f.[t]).

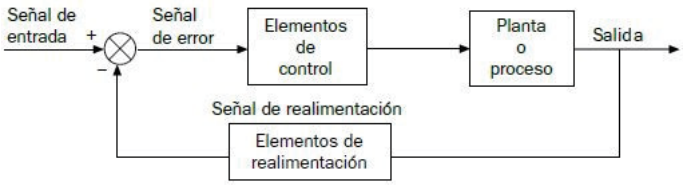


Figura 6.2 Control Lazo Cerrado (Mateo Cruz, 2019).

El control con lazo cerrado presenta las siguientes ventajas sobre el control en lazo abierto:

- corrección de las perturbaciones
- buen comportamiento incluso con incertidumbre en el modelo
- permite estabilizar procesos inestables
- tolerancia a variaciones en los parámetros

6.1.3 Estrategias de Control P, PI y PID

A continuación se estudian las estrategias de control con realimentación más comunes en la industria:

Control Proporcional (P)

El controlador de acción **proporcional** aumenta o disminuye la acción del elemento final de la regulación proporcionalmente al **error** o diferencia entre el valor de la variable controlada y el de consigna.

La señal de control o magnitud de ajuste $u(t)$ es proporcional a la señal de error $e(t)$:

$e(t) = referencia - variable\,controlada$

$u(t) = K_p \cdot e(t)$

El coeficiente de proporcionalidad K_p recibe el nombre de ganancia. El dispositivo controlado se posiciona proporcionalmente en respuesta a pequeños cambios de la variable controlada.

Entre el valor de consigna y el realmente obtenido hay siempre una diferencia o **error residual** característico de este sistema de regulación. Se puede minorar este error aumentado el coeficiente de proporcionalidad, pero se corre el riesgo de que el sistema se vuelva **inestable** (En Wikipedia, s.f.[t]).

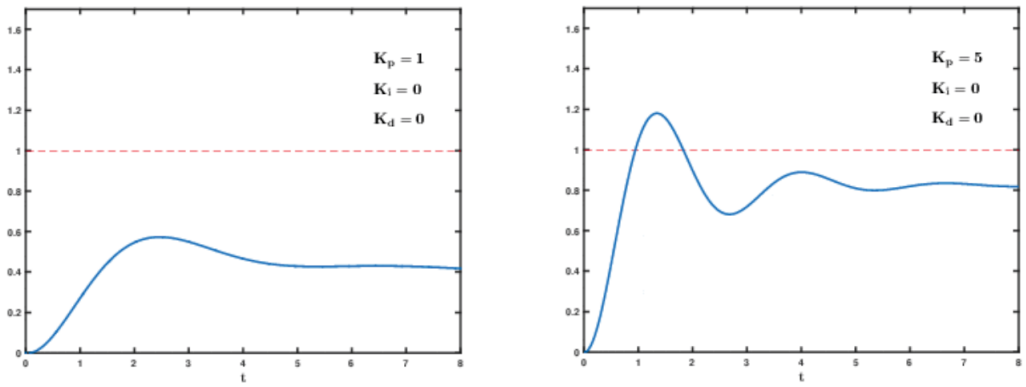


Figura 6.3 Control P (Physicsch, 2015).

Control Proporcional-Integral (PI)

La acción de control, en este caso, viene definida por la siguiente ecuación:

$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt$ donde Kp es la constante proporcional y K_i la constante integral.

Si se reemplaza $K_i = K_p/T_i$, se obtiene:

$u(t) = K_p(e(t) + \frac{1}{T_i}\int_0^t e(t)\,dt)$ donde T_i representa el tiempo de integración, el tiempo que va a tardar el término integral en compensar todo el error **acumulado**.

Esta acción de control está compuesta por la suma de la acción **proporcional** y la acción **integral**. La acción proporcional suministra al dispositivo controlado una señal para

corregir el error, siempre que este no sea nulo, y la acción integral es proporcional **al error y a su duración** y tiende a **eliminar** el error residual o error acumulado.

Con el aumento de K_i se consigue eliminar el error en menos tiempo pero para K_i grandes la parte Integral provoca **sobreoscilaciones** indeseadas en la salida del sistema (En Wikipedia, s.f.[t]).

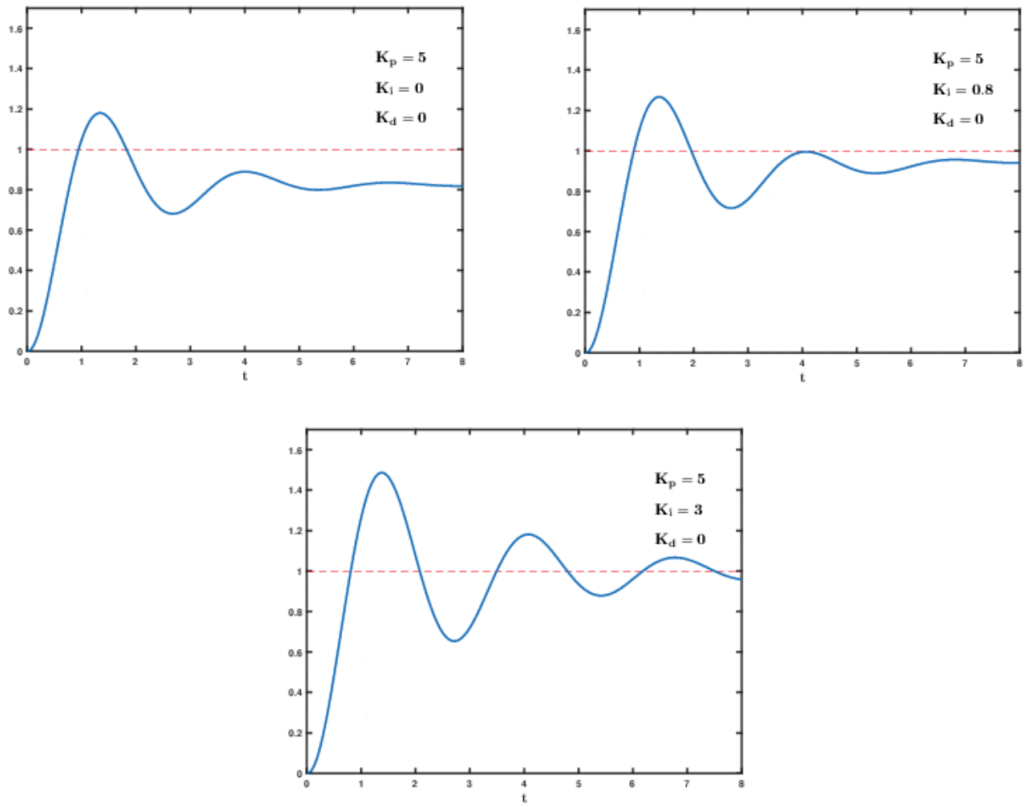


Figura 6.4 Control PI (Physicsch, 2015).

Control Proporcional-Integral-Derivativo (PID)

La acción de control, en este caso, viene definida por la siguiente ecuación:

$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}$ donde Kp es la constante proporcional y K_i la constante integral y K_d es la constante derivativa.

Si se reemplaza $K_i = K_p/T_i$, y $K_d = K_p T_d$ se obtiene:

$u(t) = K_p(e(t) + \frac{1}{T_i}\int_0^t e(t)\,dt + T_d \frac{de(t)}{dt})$ donde T_i representa el tiempo de integración, y T_d representa el tiempo de derivación, intenta predecir el valor de error en T_d segundos en el futuro, asumiendo que el bucle de control permanece sin cambios.

La acción derivativa corrige la posición de la variable operada proporcionalmente a la **velocidad de variación del error** para que, después de una variación de la carga, la variable controlada regrese lo más rápido posible a su valor de consigna.

El control proporcional con acción integral tiene el inconveniente de no tener en cuenta la tendencia del error, es decir, no es capaz de distinguir una situación en la que el error

está creciendo, de otra en la que está disminuyendo.

La acción derivativa tiene en cuenta los valores futuros del error y es capaz de distinguir los dos casos, predice el comportamiento del sistema y, por lo tanto, mejora el **tiempo de establecimiento** y la **estabilidad** del sistema (En Wikipedia, s.f.[t]).

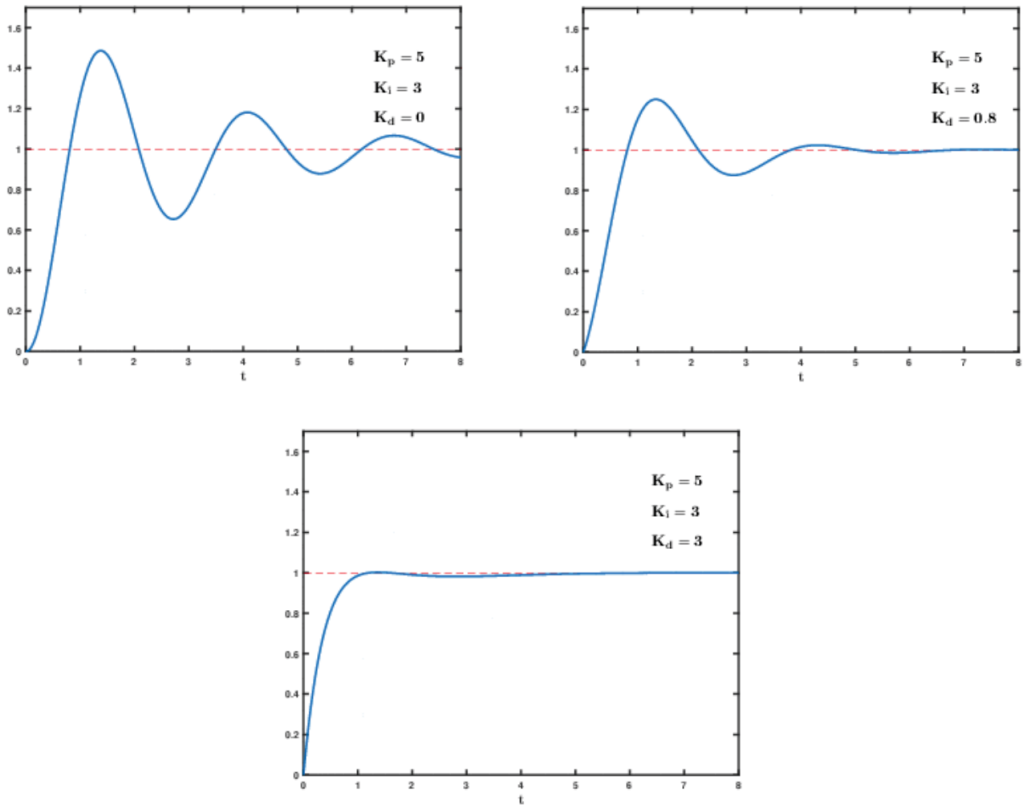


Figura 6.5 Control PID (Physicsch, 2015).

6.1.4 Automatización Industrial

Introducción

La **automatización industrial** es el uso de sistemas o elementos computarizados y electromecánicos para fines industriales.

Como una disciplina de la ingeniería más amplia que un sistema de control, abarca la instrumentación industrial, que incluye los sensores, los transmisores de campo, los sistemas de control y supervisión, los sistemas de transmisión y recolección de datos y las aplicaciones de software en tiempo real para supervisar y controlar las operaciones de plantas o procesos industriales.

El objetivo, **aumentar la producción, mejorar la calidad y evitar riesgos** para las personas. Esto puede llevar asociado una pérdida de puestos de trabajo poco cualificados, pero a cambio se generan otras necesidades de personal para diseñar, fabricar, poner en marcha y controlar estos nuevos procesos.

La ingeniería, el mantenimiento industrial y las nuevas profesiones relacionadas con la

automatización cobran entonces una nueva dimensión (En Wikipedia, s.f.[b]; Aldakin, s.f.).

- **Tecnologías que componen la Automatización Industrial:**
  - Electricidad y la electrónica industrial.
  - Neumática industrial.
  - Oleohidráulica industrial.
  - Autómatas programables.
  - Comunicaciones industriales.
  - Robótica industrial.

Sistemas de control distribuido

Un **sistema de control distribuido** está formado por varios niveles de automatización, véase figura 6.6. Los mismos se denominan:

- **nivel de campo**: donde se encuentran los sensores y actuadores
- **nivel de control**: donde se encuentran los PLCs o las Estaciones de Automatización
- **nivel de supervisión**: donde se encuentran las Estaciones de Operación y los Servidores de Proceso
- **nivel MES**: donde se encuentran PCs con software especializado para la distribución de toda la información de planta así como la generación de reportes
- **nivel ERP**: donde se encuentran igualmente PCs con software especializado para la planificación y administración de la producción de toda la industria o empresa (En Wikipedia, s.f.[b]).

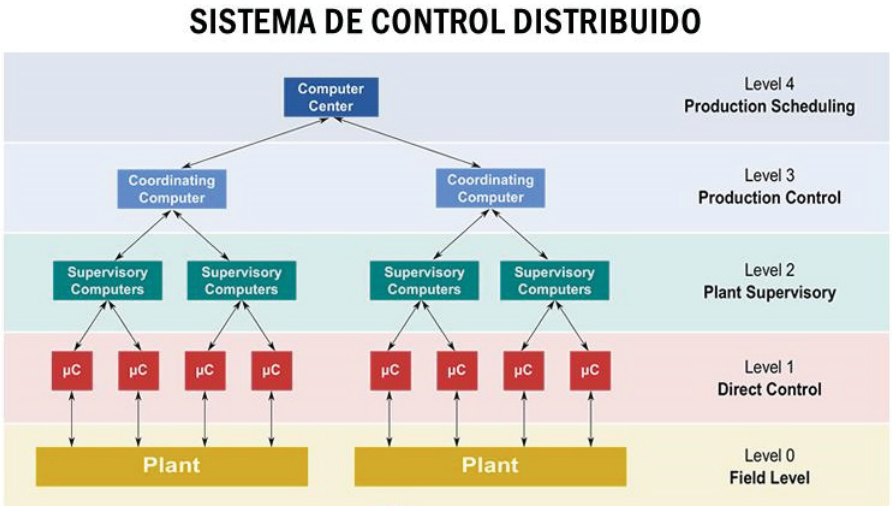


Figura 6.6 Sistema de Control Distribuido (Recursos, 2018).

Autómata Programable/Programmable Logic Controller (PLC)

Un **autómata programable**, es una computadora utilizada en automatización industrial, para automatizar procesos electromecánicos, tales como el control de la maquinaria de la fábrica en líneas de montaje.

Los PLC son utilizados en múltiples industrias y máquinas. A diferencia de las computadoras de propósito general, el PLC está diseñado para múltiples señales de entrada y de salida, rangos de temperatura ampliados, inmunidad al ruido eléctrico y resistencia a la vibración y al impacto.

Los programas para el control de funcionamiento de la máquina se suelen almacenar en copias de seguridad y/o en memorias no volátiles.

Un PLC es un ejemplo de un sistema de tiempo real, donde los resultados de salida deben ser producidos en respuesta a las condiciones de entrada dentro de un tiempo limitado, de lo contrario no producirá el resultado deseado.

los PLC no sólo controlan la lógica de funcionamiento de máquinas, plantas y procesos industriales, sino que también pueden realizar operaciones aritméticas, manejar señales analógicas para realizar estrategias de control, tales como controladores **PID** (En Wikipedia, s.f.[f]).

Supervisión, Control y Adquisición de Datos (SCADA)

SCADA es un concepto que se emplea para realizar un software para ordenadores que permite controlar y supervisar procesos industriales a distancia.

Facilita la realimentación en tiempo real con los dispositivos de campo (sensores y actuadores), y controla el proceso automáticamente.

Provee de toda la información que se genera en el proceso productivo (supervisión, control calidad, control de producción, almacenamiento de datos, etc.) y permite su gestión e intervención (En Wikipedia, s.f.[u]).

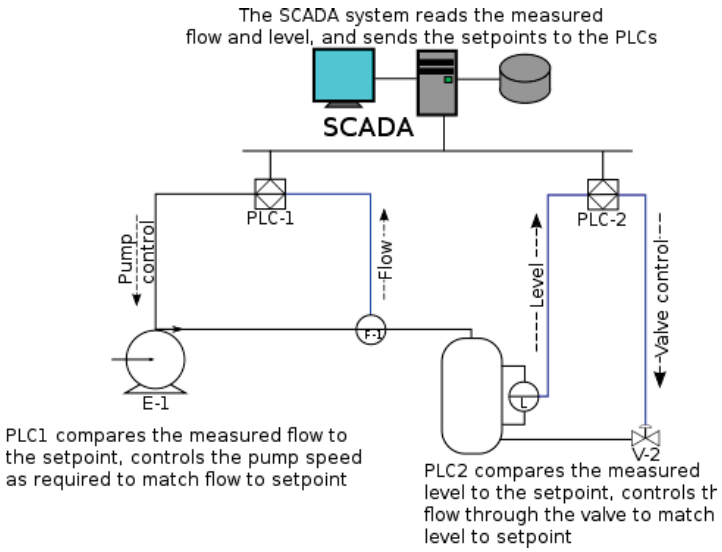


Figura 6.7 Estructura de supervisión con SCADA(En Wikipedia, s.f.[u]).

6.2 Blockchain y el Control Automático

6.2.1 Introducción

En el capítulo 2, se destacan los beneficios de implementar la tecnología Blockchain con otras tecnologías para obtener las **ventajas** y características principales de la tecnología Blockchain, confianza/fiabilidad, seguridad y verdad en datos/transacciones, todo esto reduciendo los costes y logrando la independencia de intermediarios/terceras partes.

En la industria de la Automatización Industrial se utilizan sistemas de adquisición de datos y control muy robustos, con alta tolerancia a fallos y redundancia. Todos estos sistemas actúan normalmente en red interna con sistemas redundantes y pocos puntos de acceso exteriores.

Uno de los principales riesgo de este sector de la industria es la ciberseguridad, el peligro de que un hacker acceda a la red de controladores y tome el control, puede desestabilizar fácilmente el sistema de producción atacado.

6.2.2 ¿Cómo puede ayudar Blockchain a la Automatización Industrial?

A continuación se estudian algunas soluciones Blockchain a los 3 niveles más comunes de los sistemas de control distribuido, los niveles de campo, control y supervisión.

Nivel de campo

El **nivel de campo** es donde se encuentran los sensores y actuadores, es el nivel más **vulnerable/crítico**, fallos en los sensores y en los actuadores puede desembocar en averías en las máquinas, accidentes y/o afectar a la producción.

Es muy importante que la adquisición y envío de datos sea segura, fiable y verdadera.

Como se observa en el diagrama 6.8, un ataque a las comunicaciones sensor-PLC permitiría introducir datos falsos en el PLC, al igual que un ataque a las comunicaciones PLC-Actuador, en ambos casos puede desembocar en un mal funcionamiento del actuador.

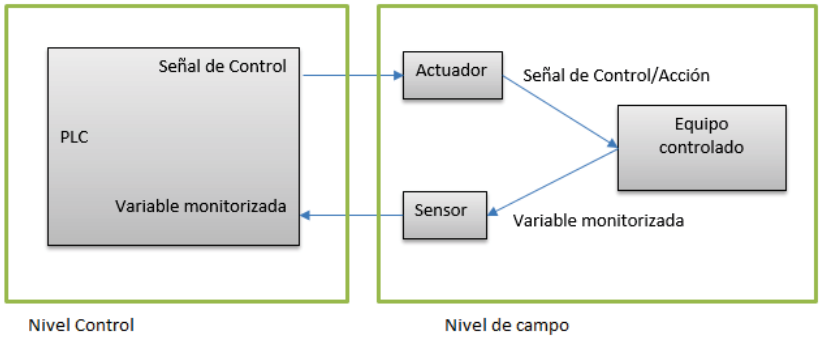


Figura 6.8 Estructura conexión nivel campo y nivel control.

Una solución Blockchain para aumentar la seguridad es utilizar la tecnología de llaves público-privadas, donde el PLC conoce la dirección pública de los sensores que controla,

y podría verificar que los datos recibidos han sido firmados por el sensor en el momento de la adquisición y no provienen de otra fuente de datos, con este método se asegura la **autoría** del dato, e igualmente para la comunicación PLC-actuador, donde el actuador solo aplica comandos firmados por el PLC.

Para asegurar además la **integridad** de esos datos los sensores, actuadores y PLCs de la red privada del sistema en cuestión se interconectan para integrar una Blockchain Local/Privada, donde sensores, actuadores y PLCs son nodos de esa Blockchain y si alguno de ellos es comprometido o sus datos se pierden la Blockchain actúa de Backup del sistema y las transacciones de datos son seguras, véase diagrama 6.9.

Un ejemplo de Blockchain que integra un sistema de tratamiento de datos similar al comentado es IOTA, un proyecto actualmente en desarrollo, para impulsar el futuro del Internet de las Cosas (IoT) con microtransacciones de datos sin comisiones y asegurando la integridad de los datos de los dispositivos; para ello usan su tecnología Tangle, basada en DAG, entre sus características principales se encuentran (iota community, 2018):

- **Altamente escalable.** A mayor actividad en la red menores son los tiempos de validación de las transacciones.
- **Bajos requisitos de recursos computacionales.** Diseñado para pequeños dispositivos, como sensores.
- **Transacciones sin coste.**
- **Transferencia segura de datos.** Todos los datos están codificados, lo que permite la transferencia segura de datos, el almacenamiento y la referencia.
- **Transacciones offline.** Los dispositivos no necesitan una conectividad perfecta.
- **Quantum Immune.** El uso de las firmas especiales IOTA es resistente a la próxima generación de computación cuántica.
- **Redes Privadas.** Permite integrar y ejecutar su tecnología Tangle en redes privadas.

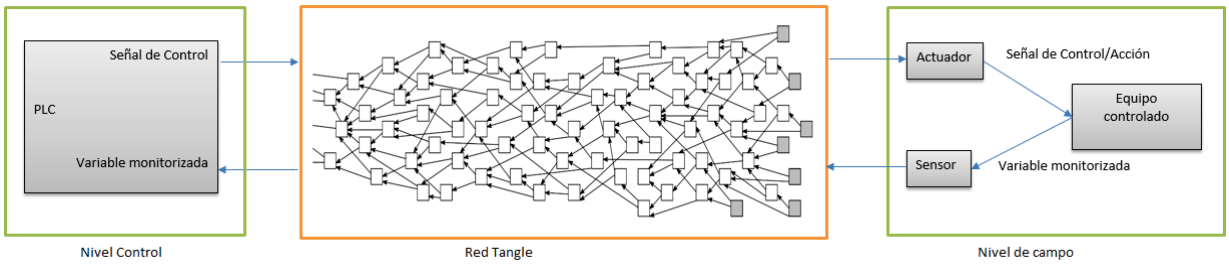


Figura 6.9 Conexión niveles campo y control mediante tecnología Tangle (Elaboración propia, 2019) (iota community, 2018).

Nivel de control

El nivel de control es donde se encuentran los PLCs o las Estaciones de Automatización, estos dispositivos se encargan de controlar los equipos bajo su gestión para que

cumplan con las consignas enviadas por el SCADA de planta que supervisa conjuntamente todos los procesos de la cadena de producción.

Al igual que en el apartado anterior, la integridad de los datos tanto aguas arriba hacia al SCADA como aguas abajo hacia el equipo controlado es un punto crítico del bucle de control, este punto crítico puede protegerse o mejorarse con proyectos como IOTA.

Además de las comunicaciones con otros dispositivos, los PLCs tienen otro punto crítico, son los encargados de ejecutar el lazo de control de primer nivel (mantener la consigna en el equipo controlado), este lazo de control suele ser un PID industrial o un sistema de control similar programado en el PLC y que se ejecuta continuamente.

Un hacker que no pueda atacar a las comunicaciones o a los datos podría intentar atacar directamente a los PLCs que controlan el proceso, por ejemplo modificando los parámetros del PID (K_p, K_i, K_d), o directamente modificando el código del software controlador.

Una solución Blockchain para este riesgo es ejecutar el programa de control en un Smart Contract en una Blockchain pública o privada.

Como se observa en la figura 6.10, los programas de control de los PLCs de la planta se han creado como Smart Contracts y se ejecutan continuamente en una Blockchain.

Cuando el Smart Contract recibe datos de la variable monitorizada y de la consigna de referencia desde el PLC, realiza el control PID y genera la señal de control que se debe aplicar sobre el sistema.

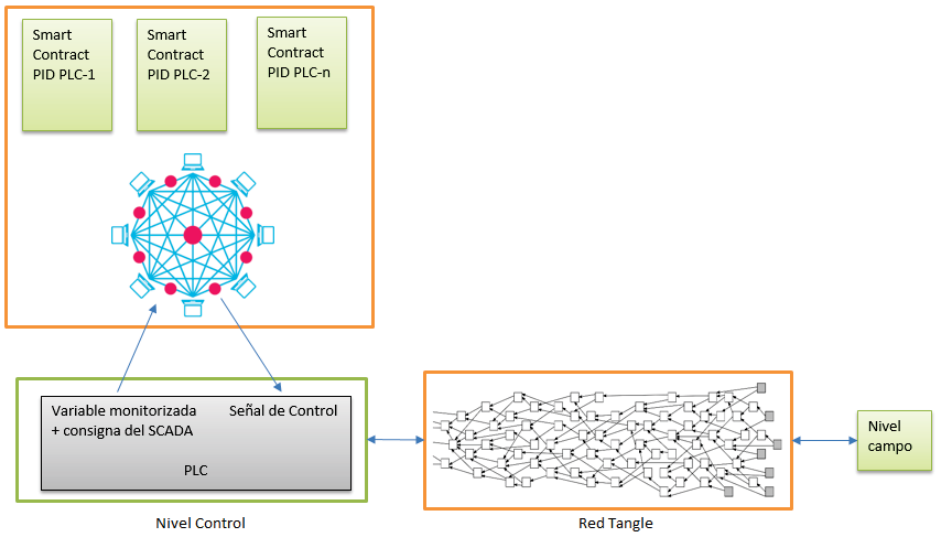


Figura 6.10 Integración de PLCs con Smart Contract (Elaboración propia, 2019)(Rosic, 2016).

Ventajas y desventajas de esta solución:

- Las principales ventajas son que una vez el programa está en la Blockchain es inmutable, no se puede modificar o hackear para que tenga un mal comportamiento, y que el código junto con todas las transacciones quedarían registrados por si es necesario

hacer auditoría en caso de algún fallo/error, por supuesto todos los datos estarían firmados y comprobada su autoría.

- La principal desventaja actualmente es que las Blockchains públicas, que son las más seguras al estar más distribuidas, son también más lentas que una Blockchain privada y generalmente tienen comisiones de transacción.
  Por ejemplo si se utilizase Etherum como solución Blockchain para este problema, el tiempo de ejecución mínimo del smart contract es de 12 segundos + el retardo de la red por bucle de control, insuficiente para un control de primer nivel en la mayoría de procesos industriales.
  Esta desventaja podrá ser solventada en el futuro por nuevos proyectos de Smart Contracts como EOS cuyo tiempo de ejecución es de 500 ms debido a su estructura de Consensus basada en DPoS que permite validar bloques mucho más rápido que PoW.

Nivel de supervisión

El **nivel de supervisión** corresponde a los sistemas de supervisión, control y adquisición de datos, en general el SCADA de planta.

En este nivel se gestiona el almacenamiento y tratamiento de datos y la aplicación de consignas a los diferentes PLCs para cumplir con los objetivos de producción.

- **Riesgos de datos:**

  Uno de los riesgos de este nivel es la pérdida de datos, normalmente se soluciona con la implementación de servidores de datos redundantes y/o backups, esta solución puede ser cara para grandes cantidades de datos con requerimientos de alta redundancia, además se corre el riesgo de tener la redundancia centralizada en un mismo punto, en caso de accidente pueden perderse el sistema de datos en producción y los backups.

  Una solución común es enviar los datos agregados a Centros de Control, pero esta solución no suele ser barata.

  La tecnología Blockchain es perfecta para almacenar datos de manera inmutable y permanentemente, conectar el SCADA de planta a una solución Blockchain de tratamiento de datos permitiría mitigar los riesgos descritos a la vez que ahorrar grandes costes en hardware.
  Unos ejemplos de proyectos Blockchain de almacenamiento de datos distribuido son Swarm, Interplanetary File Storage (IPFS), SIA, Storj.io.

  La gran ventaja de estos proyectos Blockchain frente al almacenamiento local es que el almacenamiento de archivos basado en Blockchain descentralizado es más seguro, hace que sea más difícil perder datos y promete ser más barato que las opciones de almacenamiento local o en la nube.

  Además del almacenamiento seguro de datos, la tecnología Blockchain facilita la auditoría y el control de los datos. Si todos los dispositivos de una planta están integrados en la tecnología Blockchain, una vez que el dato es generado y enviado a la red por el dispositivo que lo crea, que incluye su valor, estampa temporal y su valor de calidad y está debidamente firmado, se añade a la Blockchain y esa información es inmutable.

En comparación con los sistemas actuales donde el Administrador del sistema tiene control total de los datos, puede modificarlos y/o borrarlos sin dejar constancia del cambio.

- **Riesgos de control:**

  Al igual que en el nivel de control con los PLCs, la mayoría de SCADAs industriales realizan bucles de control a nivel de planta o de línea de producción, en general los SCADAs de planta tienen lectura de varias variables de salida generales de la planta, y permiten escribir consignas para esas variables de salida a los operadores de planta a través de un HMI (interfaz hombre máquina).

  Los riesgos en este nivel son un ciberataque sobre el hardware del SCADA con la intención de anularlo o sobre el software para modificar su programa de control, de manera que los PLCs pierdan la consigna de referencia aplicada, o un intento de suplantar la identidad del operario debido a que los sistemas de autenticación suelen ser del tipo usuario/contraseña, poco seguros si un hacker ha obtenido acceso a la Base de Datos de configuración.

  La solución Blockchain para estos riesgos podría ser ejecutar el control de la planta mediante uno o varios Smart Contracts.

  Normalmente los bucles de control de planta no son tan demandantes en velocidad de ejecución como los bucles de control de los PLCs, por tanto la velocidad de ejecución del Smart Contract (que cada vez es más reducida) no se puede considerar en este caso una gran desventaja.

  A cambio, al ejecutar el software de Control en un Smart Contract (sobretodo si es de Blockchain pública) se eliminan los riesgos de fallo en el hardware, modificación de software y el riesgo de suplantación de identidad.

  El Smart Contract puede recibir consignas directamente desde el operario, solo tramitará aquellas firmadas por su llave privada, si está conectado directamente con los PLCs no necesita el hardware del SCADA para tramitar las consignas, y el programa de control será inmutable y resistente a hackers.

  Con esta solución se traslada el punto crítico a la estabilidad de las comunicaciones, tanto en red privada en caso de implementar una Blockchain privadas, como en acceso a Internet para Blockchain públicas.

Aplicando estas soluciones Blockchains al SCADA de planta, este pasaría a ser una mera interfaz HMI para interactuar con la Blockchain de control y con la Blockchain de datos, permitiendo un hardware de SCADA más barato y en caso de fallo de este hardware no existirían riesgos de problemas de control o de pérdida de datos, véase el diagrama 6.11 que ilustra la estructura que se ha comentado.

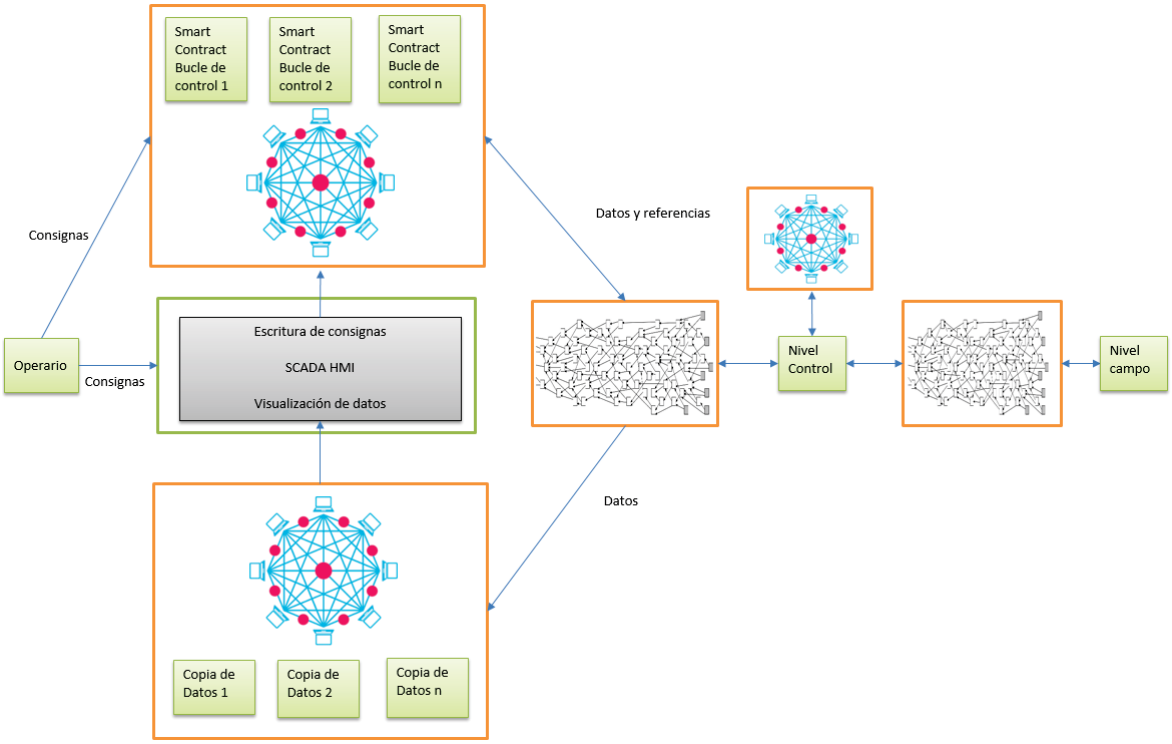


Figura 6.11 Integración de SCADA con Smart Contract y Storage Blockchain (Elaboración propia, 2019) (Rosic, 2016).

6.2.3 Conclusiones

En los apartados anteriores se han comentado algunos de los riesgos más habituales en los sistemas de automatización, los principales son los relativos a ciberseguridad, en caso de una brecha de seguridad la integridad de la planta podría estar en peligro por un mal uso de los sistemas de control y los problemas relativos al hardware, la necesidad de redundancia en estos equipos caros aumenta el precio de las soluciones SCADA y PLCs.

Se ha dado un visión teórica de cómo la tecnología Blockchain puede dar una solución a estos riesgos, y como parte de esos riesgos se transmitirían/concentrarían en la red de comunicaciones, privada si es una solución Blockchain privada, o Internet si es una solución pública.

En un supuesto de tecnología de sensores y actuadores robustos y más desarrollados en la capa de comunicaciones (objetivos del IoT), e interfaces web alojadas en Blockchains (DApps), podría llegarse a una solución Blockchain Total eliminando a todos los intermediarios (PLCs y Hardware SCADA) del proceso, e interactuando el operario directamente con los equipos de planta y con los datos a través de la solución Blockchain, de manera fiable y segura, usando el sistema de firmas para asegurar su identidad, ver figura 6.12.

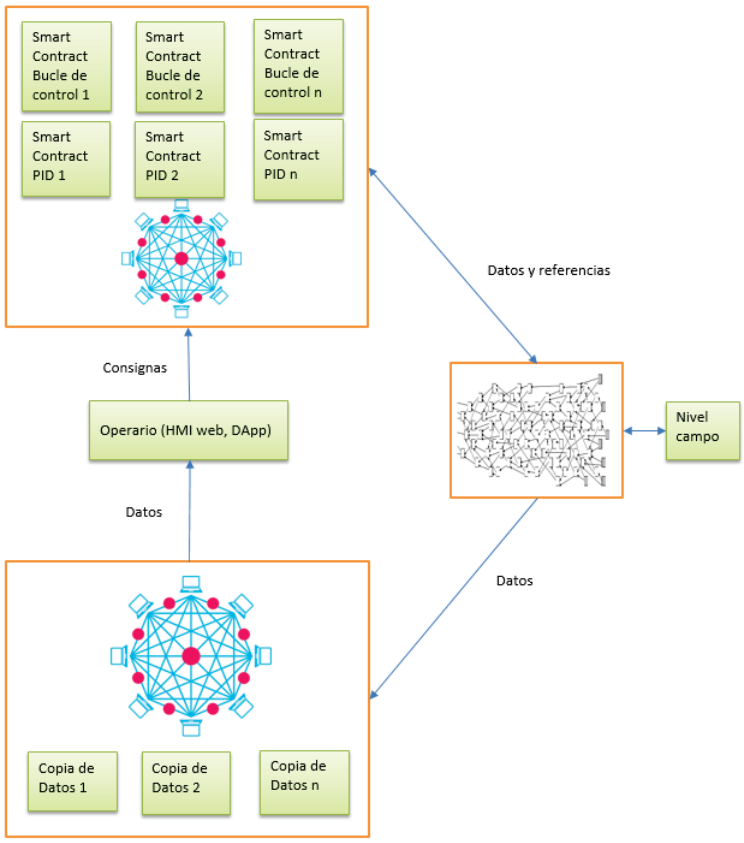


Figura 6.12 Integración Total sin dispositivos intermediarios (Elaboración propia, 2019) (Rosic, 2016).

7 Control PI en Blockchain

7.1 Estado del Arte

En el articulo **Blockchain-based Technology for Industrial Control System Cyper-Security**(Mao y Xiao, 2018) se expone la idea de aplicar tecnología Blockchain a los **Sistemas de Control Industrial** para mejorar la seguridad de la red, concluyendo que añadir tecnología Blockchain en este ámbito se consigue un sistema confiable, seguro, de alta eficiencia y bajo coste, consiguiendo redes de dispositivos IoT más seguras.

Mingyang Mao y Hong Xiao concluyen que los nodos IoT forman una red y se comunican Peer to Peer usando claves públicas y privadas, logrando una comunicación **segura** entre ellos, y debido a los mecanismos de consenso las transacciones no válidas son rechazadas, gracias a esto evitan la interacción de atacantes en la red.

Profundizando más en el tema, se encuentra el trabajo **Blockchain Technology with Applications to Distributed Control and Cooperative Robotics: A Survey** (Khan y col., 2018), donde se indica la importancia de las **decisiones distribuidas** (consenso) y el mecanismo de seguridad en la tecnología Blockchain para realizar Control Distribuido y aplicarlo al campo de la Cooperación Robótica, finalmente concluyen que a pesar de lo disruptivo, la tecnología de robots de enjambre (swarm) y los Sistemas de Control Distribuido, todavía enfrentan desafíos de privacidad y seguridad.

Blockchain facilita superar estos desafíos con su descentralización, y la configuración en red Peer to Peer. Su descentralización elimina el monopolio del Third Party y crea un espacio **seguro** en un entorno **poco confiable**, lo que lo hace ideal. Se están produciendo cambios tecnológicos e innovadores en futuro, Blockchain puede revolucionar el campo de los sistemas DCS y enjambre de robots y les proporcionará una forma más segura y ambiente infalible.

Aún no existen muchas publicaciones sobre tecnología Blockchain aplicada al campo de Control Automático y Sistemas de Control Distribuido, aunque es cada vez más fácil encontrar casos de uso de Blockchain distinto al uso como criptomoneda.

7.2 Introducción

En este capítulo se va a aplicar la tecnología Blockchain a un **sistema IoT** para utilizar las ventajas de Blockchain en mejorar los puntos débiles de estos nuevos sistemas.

El caso de uso es la climatización de una vivienda, donde exista una red IoT de sensores de temperatura y un Controlador Centralizado que actúa sobre el aire acondicionado de la vivienda.

Posibles riesgos si la red IoT de esta vivienda sufre un ataque, como se muestra en la figura 7.1:

1. Atacante suplante la identidad de los sensores, generando información falsa para confundir al Controlador Central.

2. Atacante suplante la identidad del Controlador Central y envía consignas a los actuadores del sistema con intención de causar averías y mal funcionamiento.

3. Atacante modifique los parámetros o código del Controlador Central.

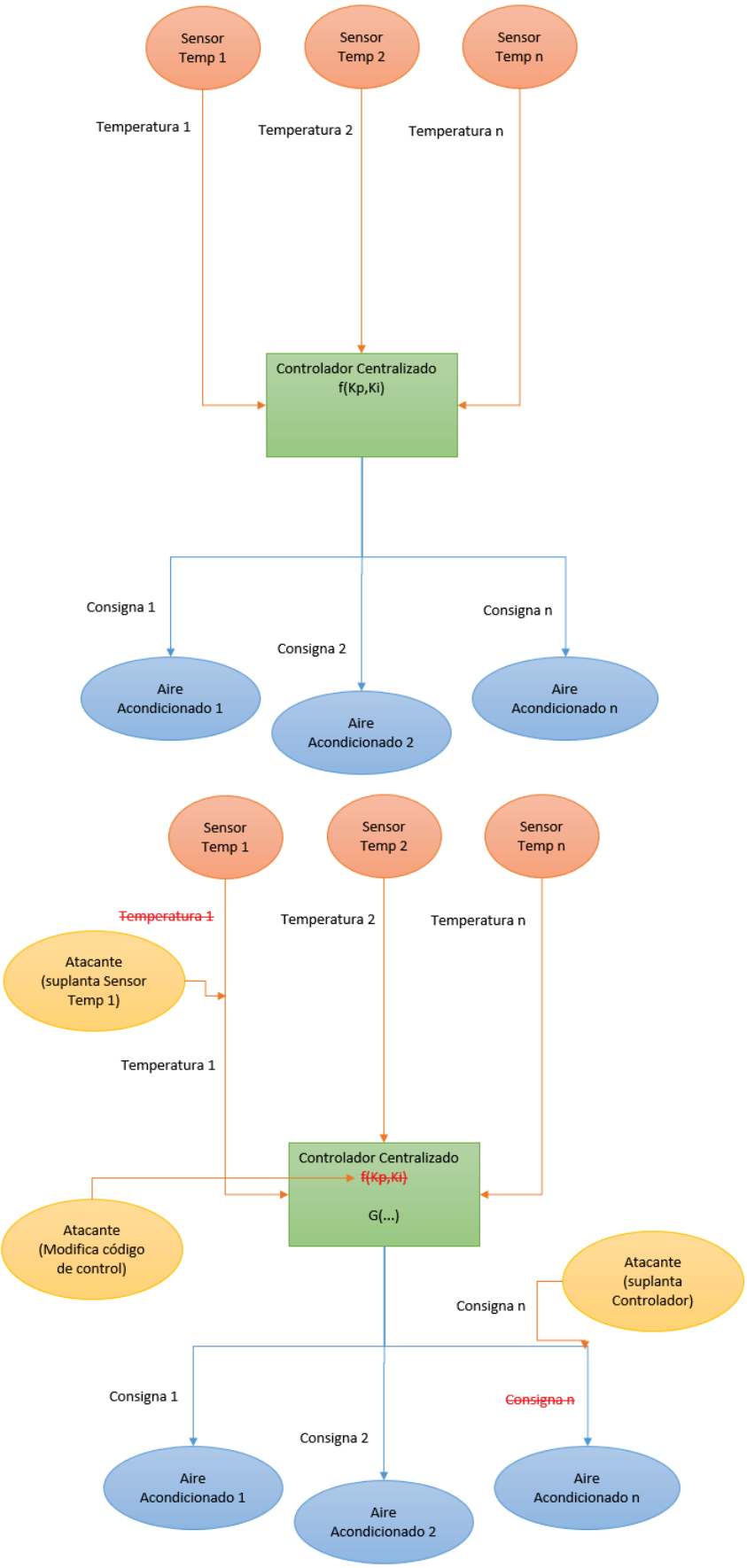


Figura 7.1 Riesgos sistema centralizado bajo ataque (Elaboración propia, 2019).

Para atacar estos riesgos, se propone las siguientes soluciones recogidas en la figura 7.2:

1. Utilizar claves públicas y privadas para **firmar** los paquetes de datos, tanto las lecturas de los sensores de temperatura como las órdenes del controlador central, con esto se logra que no se pueda introducir transacciones falsas por nodos o dispositivos ajenos a la red, solo aquellos sensores **confiables** poseen las llaves privadas para firmar transacciones.

2. Utilizar tecnología Blockchain de consenso para **validar** las transacciones y descartar las falsas, los sensores no envían datos directamente al Controlador, en su lugar alojan sus datos en la Blockchain donde son validados, y los actuadores no aplican los comandos recibidos del Controlador directamente, aplican los comandos calculados por el Smart Contract que leen de la Blockchain, si un atacante intenta modificar los datos de los sensores o las consignas del Controlador ya validados en la Blockchain, esas modificaciones serán **rechazadas**.

3. Alojar el código del Controlador en un Smart Contract en la Blockchain, de esta manera el bucle de control no podrá ser **modificado** una vez validado y sus parámetros solo podrán ser modificados por usuarios **autorizados** en el Smart Contract que firmen las transacciones con su clave privada.

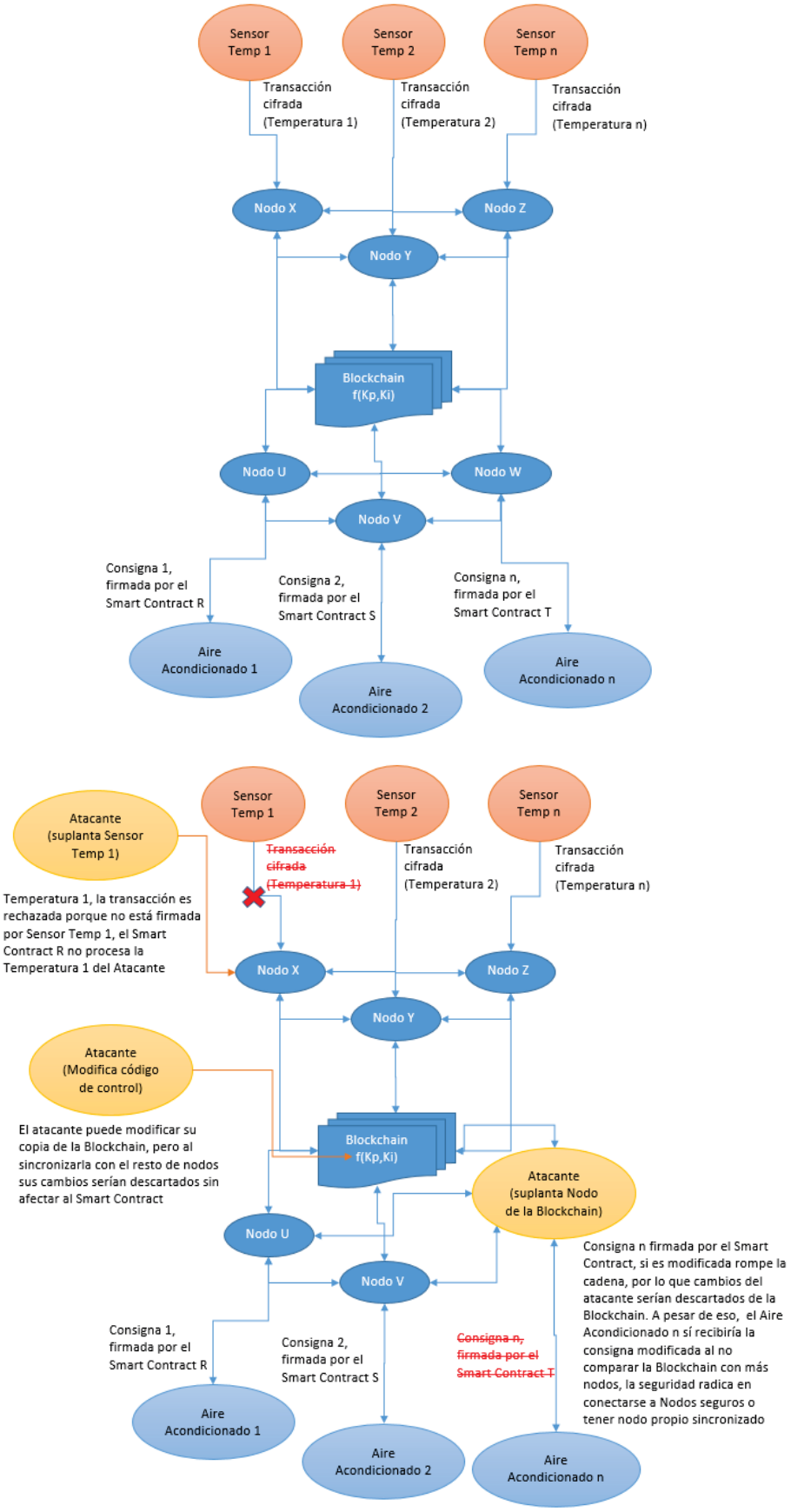


Figura 7.2 Solución Blockchain propuesta (Elaboración propia, 2019).

7.3 Objetivo, solución adoptada y elementos utilizados

7.3.1 Objetivo y solución adoptada

El objetivo es demostrar que se puede mejorar una red simple de IoT susceptible a ataques aplicando **Blockchain** a su desarrollo, para ello se utiliza un **Arduino** como dispositivo IoT, este realizará las tareas de comunicación **segura** con la Blockchain y con los elementos IoT como sensores y actuadores, y se utiliza un Smart Contract en la Blockchain para asegurar que el bucle de control es **seguro** y **confiable**.

Para simular este caso de uso se ha construido un sistema de control de temperatura con ventiladores y un sensor de temperatura, se controlan los ventiladores con un Arduino que aplica las consignas **leídas** de la Blockchain y calculadas y escritas por un Smart Contract en la Blockchain de Ethereum, este Smart Contract recibirá las medidas de temperatura escritas por el Arduino en la Blockchain para realizar el cálculo de las consignas como se observa en el diagrama 7.3.

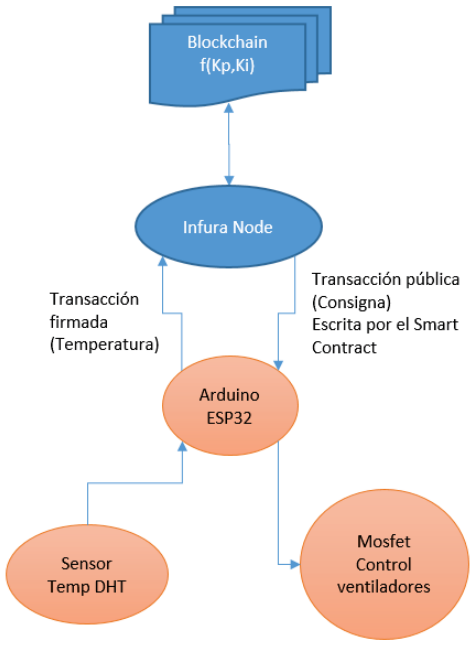


Figura 7.3 Solución adoptada (Elaboración propia, 2019).

7.3.2 Elementos Hardware

- **Arduino con adaptador Wi-Fi ESP32:**

  ESP32 es un único chip combinado de 2,4 GHz con Wi-Fi y Bluetooth diseñado con el TSMC de ultra baja potencia de 40 nm.
  Está diseñado para lograr el mejor rendimiento de potencia y RF, mostrando robustez, versatilidad y fiabilidad en una amplia variedad de aplicaciones y escenarios.
  ESP32 está diseñado para aplicaciones móviles, wearables y dispositivos de Internet de las cosas (IoT) (En Espressif Systems, s.f.).

- **Ventiladores de 5V DC y 0.2A Pi-Fan**
- **Sensor de Temperatura Arduino (AM2303) DHT22:**

  El sensor AM2302 es la versión cableada y encapsulada en un cuerpo plástico del DHT22, sencillo de usar con Arduino y Raspberry Pi.
- **Mosfet NPN TIP122G**

7.3.3 Elementos Software

- **Arduino IDE:**

  El software Arduino de código abierto (IDE) permite escribir código y subirlo a la placa Arduino.
  Se ejecuta en Windows, Mac OS X y Linux.
  El entorno está escrito en Java y se basa en software de código abierto.
- **Ethereum Blockchain - Ropsten:**

  La red de prueba de Ropsten (testnet) se usa esencialmente como un entorno de pruebas antes de llevar su código a la red principal de Ethereum (mainnet).
  A diferencia de la red principal, escribir en la red de prueba es gratis (Dakota Quint, s.f.).

  Lo más destacado de las redes de prueba de Ethereum:

  - Permite implementar código no confiable en la red de prueba en lugar de hacerlo en la red principal.
  - Es prácticamente gratuito (a diferencia de la red principal).
  - Está sujeto a cambios.
- **JSON-RPC:**

  JSON es un formato ligero de intercambio de datos. Puede representar números, cadenas, secuencias ordenadas de valores y colecciones de pares de nombre / valor.
  JSON-RPC es un protocolo de llamada a procedimiento remoto (RPC) sin estado y ligero.
  Principalmente, esta especificación define varias estructuras de datos y las reglas en torno a su procesamiento. Es independiente del transporte, a través de sockets, a través de HTTP o en muchos entornos de transmisión de mensajes. Utiliza JSON (RFC 4627) como formato de datos (Felix Lange, s.f.).
- **Infura node:**

  Infura es un clúster alojado de nodos de Ethereum que permite a sus usuarios ejecutar su aplicación de forma segura y fiable sin necesitar que configuren su propio nodo completo o wallet de Ethereum (Tim Coulter, s.f.).
- **Truffle development environment:**

  Truffle es un entorno de desarrollo (proporciona una herramienta de línea de comandos para compilar, implementar y probar Smart Contracts), de framework (proporciona varios paquetes para facilitar la escritura de pruebas, código de implementación, clientes de compilación, etc.) y de canalización de activos (permite publicar

paquetes y usar paquetes publicados por otros) para construir DApps basados en Ethereum (Narayan Prusty, 2017) .

- **Atom text and source code editor:**

  Atom es un editor de texto y código fuente gratuito y de código abierto para macOS, Linux y Microsoft Windows con soporte para complementos escritos en Node.js, e integrado en Git Control, desarrollado por GitHub.
  Atom es una aplicación de escritorio creada con tecnologías web. La mayoría de los paquetes que tienen licencias de software libre y son construidos y mantenidos por la comunidad.
  Atom se basa en Electron (anteriormente conocido como Atom Shell), un marco que permite aplicaciones de escritorio multiplataforma que utilizan Chromium y Node.js. Está escrito en CoffeeScript y Less (En Wikipedia, s.f.[a]) .

- **Solidity:**

  Solidity es un lenguaje de programación orientado a objetos para escribir Smart Contracts y se utiliza para implementarlos en varias plataformas Blockchain, especialmente Ethereum.
  Fue desarrollado por Gavin Wood, Christian Reitwiessner, Alex Beregszaszi, Liana Husikyan, Yoichi Hirai y varios antiguos colaboradores principales de Ethereum (En Wikipedia, s.f.[w]) .

- **NodeJs y Webserver:**

  Node.js es un entorno multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.
  Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.
  Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent.
  Node.js es similar en su propósito a Twisted o Tornado de Python, Perl Object Environment de Perl, libevent o libev de C, EventMachine de Ruby, vibe.d de D y Java EE de Java existe Apache MINA, Netty, Akka, Vert.x, Grizzly o Xsocket. Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor. Node.js implementa algunas especificaciones de CommonJS. Node.js incluye un entorno REPL para depuración interactiva (En Wikipedia, s.f.[q]).

- **Metamask:**

  MetaMask permite ejecutar Ethereum dApps directamente en el navegador sin ejecutar un nodo Ethereum completo, incluye una “caja fuerte segura” de identidad, que proporciona una interfaz de usuario para administrar sus identidades/llaves privadas en diferentes sitios y firmar transacciones de Blockchain.
  Se puede instalar el complemento MetaMask en Chrome, Firefox, Opera y el nuevo navegador Brave (En Metamask, s.f.).

7.4 Diseño y montaje

El objetivo es utilizar el Arduino ESP32 como pasarela entre el Smart Contract de un Controlador PI alojado en Ethereum y los dispositivos IoT (sensor de temperatura y ventiladores), para lograr esto se han realizado las siguientes tareas:

1. Montaje del circuito de lectura del sensor de temperatura DHT22
2. Montaje del circuito de control de los ventiladores
3. Desarrollo del código en Arduino para leer el sensor de temperatura, escribir la lectura en el Smart Contract, leer la señal de control calculada por el Smart Contract y aplicarla en el circuito de los ventiladores
4. Desarrollo del código del Smart Contract del Controlador PI en la Blockchain de Ethereum
5. Desarrollo de una interfaz web para leer y escribir en el Smart Contract los parámetros de configuración del Controlador PI

7.4.1 Circuito de lectura del sensor de temperatura DHT22

El sensor de temperatura DHT22 adaptado para Arduino, mide temperaturas de entre -40ºC y 80ºC con una precisión de +-0.5ºC y consta de 3 pines:

- **Pin 1:** Alimentación
- **Pin 2:** Transferencia de datos
- **Pin 3:** Conexión a tierra

La información es transmitida por el Pin 2 de Datos del DHT22 como indica el Datasheet (En sparkfun.com, s.f.) y se lee a través de un Pin Digital de entrada de Arduino.

Para su lectura se ha realizado el montaje de la figura 7.4, diagrama generado con circuito.io (En circuito.io, s.f.):

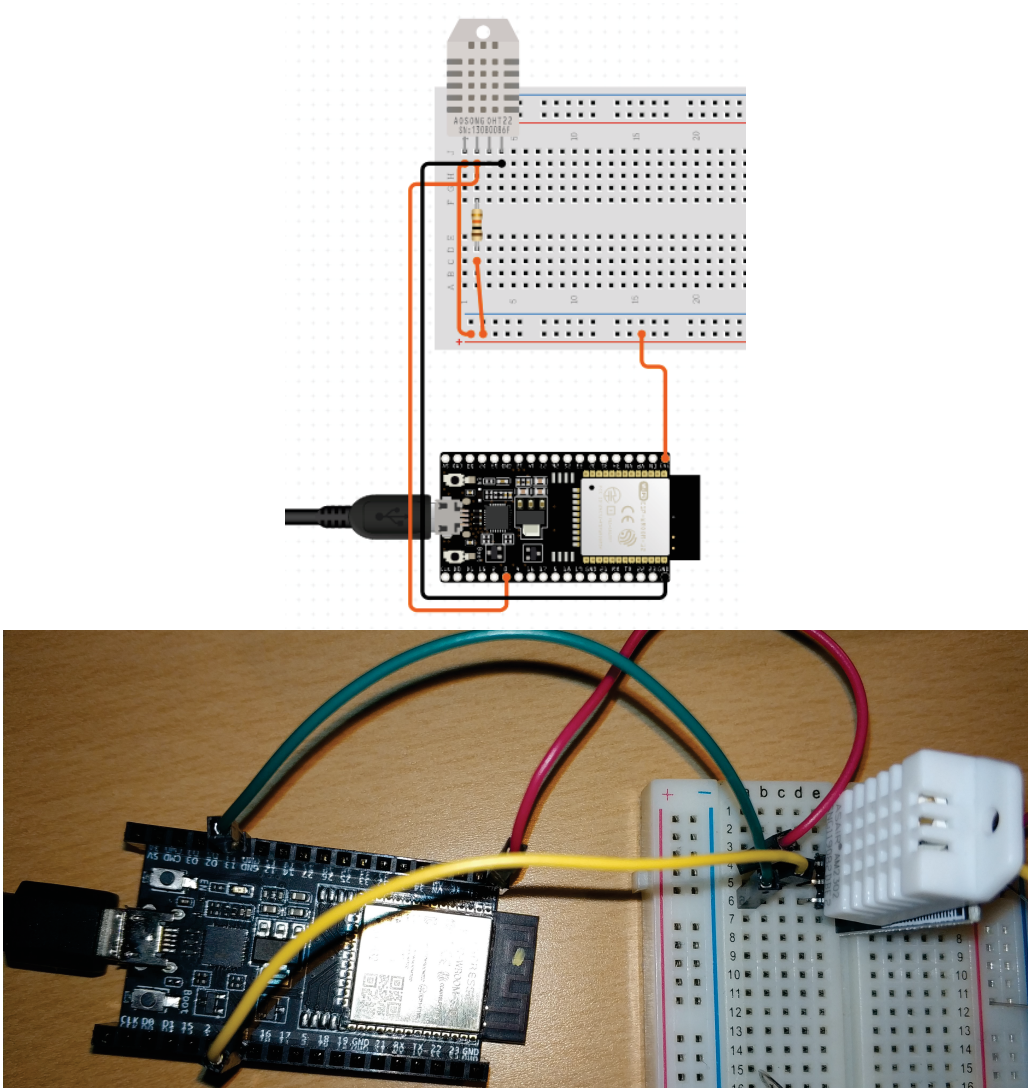


Figura 7.4 Diagrama y resultado para lectura sensor de temperatura.

7.4.2 Montaje del circuito de control de los ventiladores

Se han utilizado dos ventiladores de 5V y 0.2A. Debido a que el Arduino ESP32 funciona a 3.3V se monta un circuito para alimentar los ventiladores y otro circuito para controlarlos aplicando una señal PWM sobre un Mosfet que actúa como interruptor y que permite controlar por tanto la alimentación de los ventiladores, como se observa en la figura 7.5.

Se puede comprobar el resultado del montaje final en la figura 7.6.

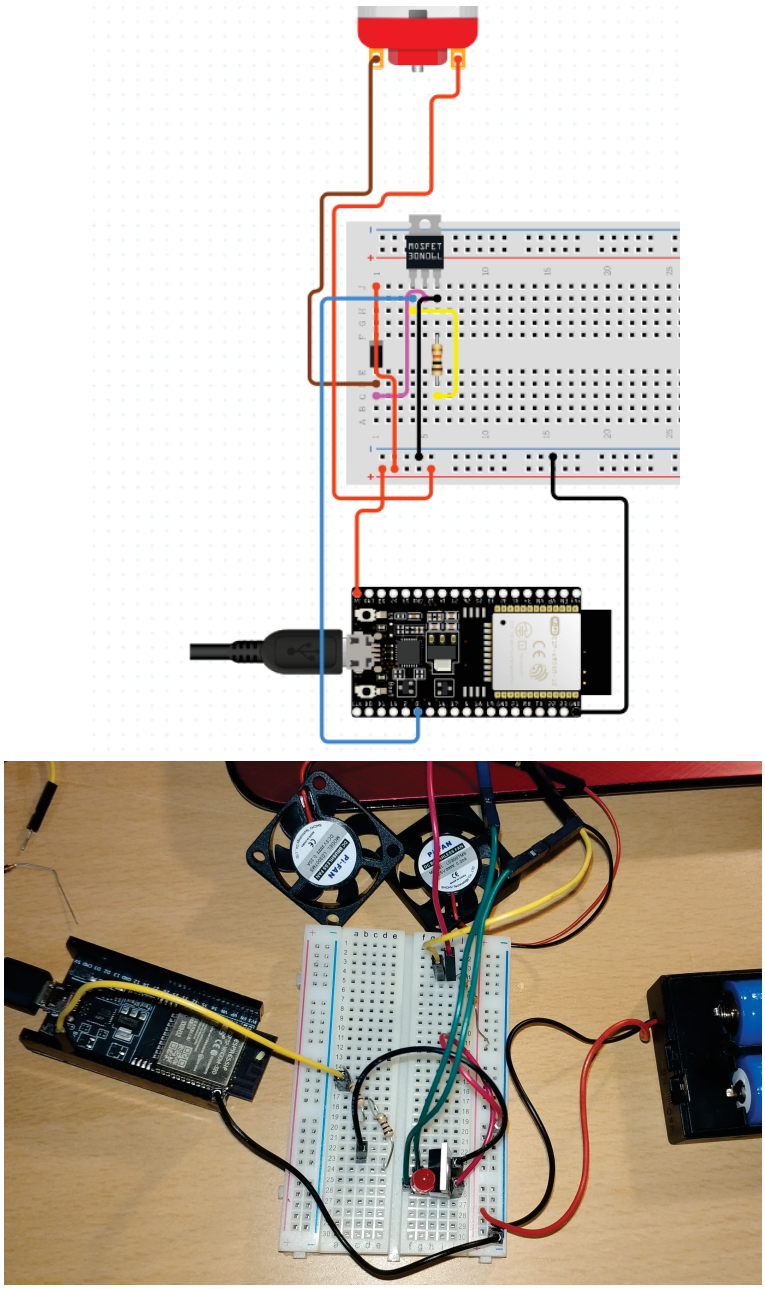


Figura 7.5 Diagrama y resultado para alimentación y control de ventiladores.

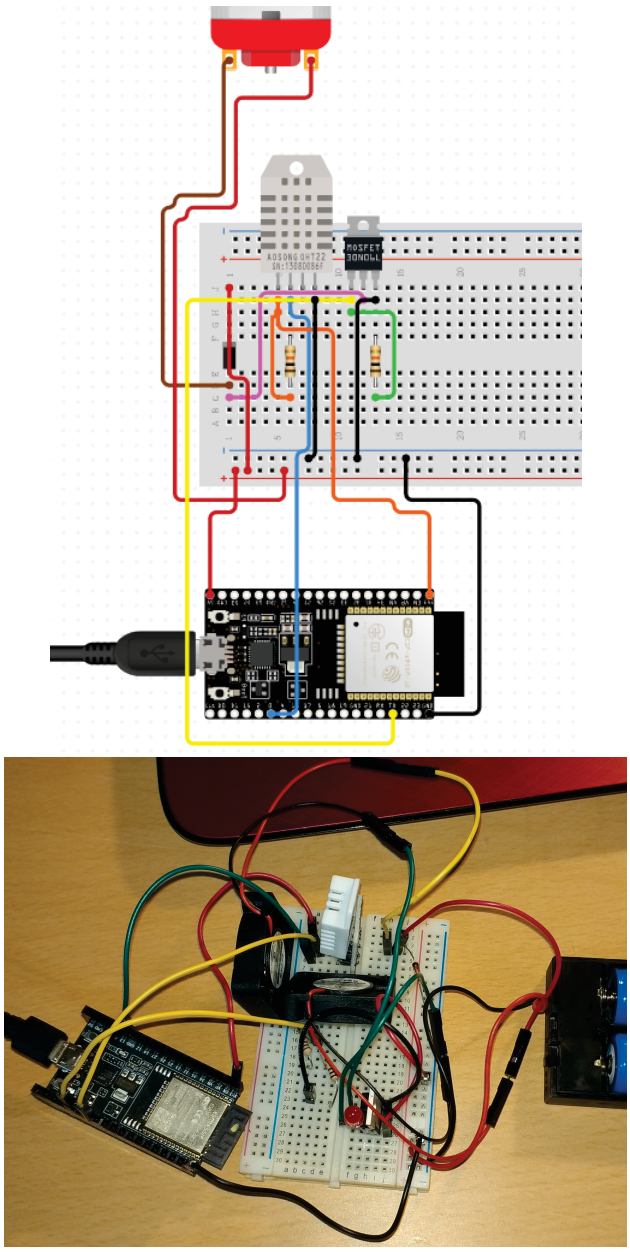


Figura 7.6 Diagrama y resultado montaje final.

7.4.3 Desarrollo del código en Arduino

Arduino se utiliza como mera **pasarela** entre Smart Contract, sensores y actuadores.

Para ello se han utilizado las librerías **Web3Arduino** (kopanitsa, s.f.) y **Arduino Wi-Fi**, que permiten al Arduino ESP32 **conectarse** a un nodo Blockchain, **firmar** transacciones y **enviarlas**, **obtener** información de los Smart Contracts, y al mismo tiempo, leer del sensor de temperatura con la librería **DHT** y actuar sobre los ventiladores.

Entrando más en profundidad en el código desarrollado y compilado usando el entorno de Arduino IDE:

```
#include "Arduino.h"
#include <ArduinoJson.h>
#include <WiFi.h>
#include <Web3.h>
#include <Contract.h>
#include "DHT.h"
/*Archivo de configuración Conf1.h, incluye información protegida:
const char* ENV_SSID
const char* ENV_WIFI_KEY
const char* INFURA_HOST
const char* INFURA_PATH
const uint8_t privateKey[]*/
#include <Conf1.h>


//Configuración librería sensor de Temp
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define USE_SERIAL Serial

//Configuración librerías Web3 y Contract
string host = INFURA_HOST;
string path = INFURA_PATH;
string address="0xB01159a43cC4Cc11D9751Ae12C340B5dD493ac35";
string contract_address="0x916E54853a6CAe90b918adE5496B63Df7fF0d5d5";

//Crea entidad Web3
Web3 web3(&host, &path);

//Define el Smart Contract
Contract contract(&web3, &contract_address);

#define FanPin 2 //Pin del ventilador
// Configuración PWM
const int freq = 500;
const int ledChannel = 0;
const int resolution = 8; //Resolution 8=0-256, 10, 12, 15
// Inicialización de la Referencia
int Ref=0;
float Reffloat=0;
// Inicialización de la temperatura
float temp=0;
int tempAdaptada=0;
// Inicialización de la señal de control
int u=0;
```

```
int uAdap=0;
// Inicialización del Nonce
uint32_t GeneralNonceVal = 0;
uint32_t nonceVal = 0;

void setup() {

Serial.begin(9600);
pinMode(FanPin, OUTPUT);
ledcSetup(ledChannel, freq, resolution);
ledcAttachPin(FanPin, ledChannel);

//Inicia comunicación con sensor de Temp
dht.begin();
//Conectar a la red Wi-Fi
connectToNetwork();

strcpy(contract.options.from,"0x916E54853a6CAe90b918adE5496B63Df7fF0d5d5");
strcpy(contract.options.gasPrice,"200000000000000");
GeneralNonceVal=(uint32_t)web3.EthGetTransactionCount(&address);

}

void loop() {
for(uint8_t t = 5; t > 0; t--) {
         delay(1000);
}

//leer Reference;
Ref= getReference();
Reffloat=float(Ref)/100;

//Leer sensor de temperatura
temp=dht.readTemperature();

//Adaptar la temperatura
tempAdaptada=temp*100;

//Enviar la temperatura y realizar el bucle de control
GeneralNonceVal=setControlSignalCalc(tempAdaptada,GeneralNonceVal);

//Leer la señal de Control a aplicar
u = getControlSignal(GeneralNonceVal);

//Adaptar señal de control
uAdap=u/100;
```

```
//parar el PWM
ledcWrite(ledChannel, 0);
delay(2000);

//Escribir como PWM la señal de control al ventilador
ledcWrite(ledChannel, uAdap);

//enviar los datos por puerto serie
Serial.printf("%2.2f,%2.2f,%d\n",Reffloat,temp,uAdap);

}
```

El código comienza definiendo las variables, con la función *Web3* $web3(host,path)$ se crea una entidad **Web3** apuntando al nodo remoto, en este caso un nodo en **Infura**, a continuación se define una entidad **Contract** $Contract\ contract(web3,contractaddress)$ donde se indican los parámetros de Smart Contract con el que se quiere comunicar.

Una vez definidas todas las variables, se inicia el bloque de **configuración** que arranca la comunicación con el sensor de temperatura, inicializa el PWM y conecta con la red Wi-Fi.

Al terminar la configuración, inicia el bloque "**Loop**", que se ejecutará en bucle, y que consta de 6 pasos:

1. Leer el valor de referencia de temperatura que está escrito en la Blockchain en el Smart Contract.
   Esta lectura se realiza meramente para exportarla posteriormente por puerto serie, que es el método que se ha decidido usar para adquirir los datos, a nivel de funcionamiento este paso no sería necesario.
   La referencia se lee como un entero de temperatura con 2 decimales, es necesario escalarlo por 100 en una variable flotante para almacenar los decimales.

2. Leer el valor de temperatura del sensor.
   Esta lectura se recibe como un flotante con 1 decimal, se multiplica por 100 para enviarlo como un entero con resolución de 2 decimales.

3. Enviar la señal de temperatura leída al Smart Contract.
   Esta acción provoca que al recibir el Smart Contract un nuevo valor de temperatura realice el cálculo de la nueva señal de control.

4. Leer el valor de la señal de control calculado por el Smart Contract.
   Adaptar esa señal que se recibe como un entero con 2 decimales entre 0 y 25500 dividiéndola por 100.

5. Escribir la nueva señal de control en el PWM que controla los ventiladores.

6. Escribir las señales de referencia, temperatura y señal de control por puerto serie para su posterior análisis.

Funciones usadas en Arduino

1. **connectToNetwork:**

   Función que conecta el Arduino ESP32 a la red Wi-Fi que se indica en el fichero *Conf*1.*h*, utiliza la librería *Wifi.h*.

```
void connectToNetwork() {

WiFi.begin(ENV_SSID, ENV_WIFI_KEY);
while (WiFi.status() != WL_CONNECTED){
    delay(1000);
    Serial.println("Establishing connection to WiFi..");
 }
}
```

2. **setControlSignalCalc:**

   Función que escribe la temperatura leída en el Smart Contract.

   Primero espera a que la última transacción enviada al Smart Contract se haya registrado/validado, a continuación genera una transacción apuntando a la función *setControlSignalCalc*(*int*256) del Smart Contract utilizando el parámetro de la temperatura leída y para finalizar firma esa transacción y la envía utilizando el protocolo JSON-RPC.

```
uint32_t setControlSignalCalc(int temp,uint32_t GeneralNonceVal) {

int t=0;
uint32_t nonceVal = (uint32_t)web3.EthGetTransactionCount(&address);
//Comprueba que se ha ejecutado la transacción anterior
    while (nonceVal!= GeneralNonceVal&&t<10) {
        // Serial.println("nonceVal es distinto a GeneralNonceVal");
        // Serial.println(nonceVal);
        // Serial.println(GeneralNonceVal);
  delay(1000);
nonceVal = (uint32_t)web3.EthGetTransactionCount(&address);
     t++;
  }

// Ejecuta la transacción
    contract.SetPrivateKey(privateKey);
    nonceVal = (uint32_t)web3.EthGetTransactionCount(&address);
    uint32_t gasPriceVal = 1041006540;
    uint32_t  gasLimitVal = 3000000;
    string toStr = contract_address;
    string valueStr = "0x00";
//func llama a la función específica en el Smart Contract
    string func = "setControlSignalCalc(int256)";
    string p = contract.SetupContractData(&func, temp);
```

```
    string result = contract.SendTransaction(nonceVal, gasPriceVal,
        gasLimitVal, &toStr, &valueStr, &p);
//Actualiza el Nonce para que el siguiente proceso espere
    nonceVal=nonceVal+1;
    return nonceVal;
}
```

3. **getControlSignal:**

   Función que lee la señal de control calculada por el Smart Contract.

   Primero espera a que se haya ejecutado/validado la última transacción pendiente en el Smart Contract, una vez comprobado que el Nonce es el último, pide el valor de la señal de control llamando a la función *getControlSignal*() del Smart Contract.

```
int getControlSignal(uint32_t GeneralNonceVal) {
int t=0;
uint32_t nonceVal = (uint32_t)web3.EthGetTransactionCount(&address);
/*Comprueba que se ha ejecutado la transacción anterior en la Blockchain
para asegurarse que lee la última señal de control calculada*/
    while (nonceVal!= GeneralNonceVal&&t<10) {
        // Serial.println("nonceVal es distinto a GeneralNonceVal");
        // Serial.println(nonceVal);
        // Serial.println(GeneralNonceVal);
  delay(1000);
nonceVal = (uint32_t)web3.EthGetTransactionCount(&address);
     t++;
  }
//Comienza la petición de parámetro
    contract.options.gas = 5000000;
    string func = "getControlSignal()";
    string param = contract.SetupContractData(&func);
        string result = contract.Call(&param);
/*Convierte la respuesta de la Blockchain al parámetro deseado
La respuesta viene en formato Json ya que usa JSON-RPC Protocol*/
        int resultInt=deserialJson(result);
    return resultInt;
}
```

4. **getReference:**

   Similar a la función *getControlSignal*, esta función pide al Smart Contract el último valor de referencia/setpoint que hay recibido.

```
int getReference() {
//Comienza la petición de parámetro
        contract.options.gas = 5000000;
        string func = "getReference()";
        string param = contract.SetupContractData(&func);
        string result = contract.Call(&param);
```

```
        int resultInt=deserialJson(result);
            return resultInt;
    }
```

5. **deserialJson:**

   Función que extrae los valores de la respuesta JSON que se recibe del Smart Contract al invocar sus funciones.

```
int deserialJson(string result){
const size_t capacity = JSON_OBJECT_SIZE(6) + JSON_ARRAY_SIZE(4) + 120;
DynamicJsonDocument doc(capacity);
// Deserialize the JSON document
DeserializationError error = deserializeJson(doc, result);
// Test if parsing succeeds.
   if (error) {
   Serial.print(F("deserializeJson() failed: "));
   Serial.println(error.c_str());
   }

//Extrae de la respuesta JSON el contenido del campo "result"
   const char* resultjson = doc["result"];
   int resultjsonint = strtol(resultjson, 0, 16);
   return resultjsonint;
}
```

7.4.4 Desarrollo del código del Smart Contract

Se ha desarrollado un Smart Contract muy sencillo para la Blockchain de Ethereum usando el lenguaje **Solidity**, a continuación se muestra el código para analizarlo con más profundidad:

```
pragma solidity ^0.5.0;

contract PIController {
//Definición de variables
  int Reference;
  int Kp;
  int Ki;
  int InputSignal;
  int ControlSignal;
  int ControlSignalAdap;
  int error;
  int error_1;
//Definición de eventos (para usarlos en la interfaz web)
  event RefEvent(
         int Reference
      );
  event KpEvent(
```

```
        int Kp
    );
  event KiEvent(
        int Ki
    );
  event ControlSignalEvent(
        int X,
        int ControlSignalAdap,
        int error,
        int error_1
    );
  event setError_1Event(
        int error_1
    );
//Definición de constructor con configuración inicial
  constructor() public {
    Reference = 2000;
    Kp = 3;
    Ki = 0;
    InputSignal=0;
    ControlSignal=0;
    ControlSignalAdap=0;
    error= 0;
    error_1=0;
  }
//Función que modifica la Referencia
  function setReference(int R) public returns(int) {
  Reference=R;
  emit RefEvent(R);
  }
//Función que modifica el parámetro Kp
  function setParameterKp(int _Kp) public returns(int) {
  Kp=_Kp;
  emit KpEvent(Kp);
  }
//Función que modifica el parámetro Ki
  function setParameterKi(int _Ki) public returns(int) {
  Ki= _Ki;
  emit KiEvent(Ki);
  }
//Función que modifica el error acumulado
  function setError_1(int _error_1) public returns(int) {
  error_1= _error_1;
  emit setError_1Event(error_1);
  }
/*Función que ejecuta el cálculo de la señal de control al recibir
un nuevo valor de temperatura*/
```

```
  function setControlSignalCalc(int X) public returns(int) {
  InputSignal=X;
//Calcula el error
  error=Reference-InputSignal;
//Calcula la señal de control
  ControlSignal=Kp*error+Ki*error_1;
//Actualiza el error acumulado
  error_1=error_1+error;
/*Comprueba que la señal de control está dentro de los rangos
establecidos equivalente a los límites de saturación*/
  if (ControlSignal>2550) {
    ControlSignal=2550;
  }
  if (ControlSignal<-2550) {
    ControlSignal=-2550;
  }
//Adaptación de los rangos de salida [-2550,2550] a [0,25500]
 ControlSignalAdap=5*(ControlSignal+2550);
  emit ControlSignalEvent(X,ControlSignalAdap,error, error_1);
  }
//Función que devuelve el valor de la señal de control
  function getControlSignal() public view returns (int) {
    return (ControlSignalAdap);
  }
//Función que devuelve el valor de la Kp
  function getKp() public view returns (int) {
    return (Kp);
  }
//Función que devuelve el valor de la Ki
  function getKi() public view returns (int) {
    return (Ki);
  }
//Función que devuelve el valor de la Referencia
  function getReference() public view returns (int) {
    return (Reference);
  }
//Función que devuelve el valor de la señal de entrada
  function getInputSignal() public view returns (int) {
    return (InputSignal);
  }
//Función que devuelve el valor del error
  function getError() public view returns (int) {
    return (error);
  }
//Función que devuelve el valor del error calculado
  function getError1() public view returns (int) {
    return (error_1);
```

```
  }
}
```

En el Smart Contract se diferencian 3 tipos de estructuras:

1. **Funciones "Set":** Son funciones que cambian el estado de una variable, para invocarlas es necesario enviar una transacción firmada, y utiliza "Gas" que es una medida del consumo necesario para ejecutar esa función por el nodo que mine esa transacción, por tanto es necesario pagar con Ethereum para cambiar el estado de una variable en la Blockchain pública de Ethereum.

2. **Funciones "Get":** Son funciones que permiten leer el Smart Contract, no incurren en ningún coste.

3. **Funciones "Event":** Son funciones que monitorizan las variables y generan un evento cuando cambia alguna variable monitorizada, con estas funciones se evita el realizar continuas peticiones a la Blockchain y en este proyecto se usarán para capturar con la interfaz web los cambios de las variables.

7.4.5 Desarrollo de una interfaz web

Para poder interactuar con el Smart Contract de una forma más ágil que usando las funciones de Arduino, se ha desarrollado una **interfaz web** para modificar los principales **parámetros** del Controlador PI y monitorizar todas las variables de interés involucradas.

Se ha hecho uso de la librería **Web3.js** y la Extensión web **Metamask** que permite interactuar con los Smart Contracts de la Blockchain de Ethereum, este código es más extenso que los anteriores, se van a repasar los puntos más importantes relacionados con Blockchain y no con creación de interfaces web:

```
/*Se indica el nodo al que se conecta, en nuestro caso
esta información la provee Metamask que es el current Provideer*/
web3 = new Web3(web3.currentProvider);

/*Se crea una variable de tipo "contract" con la información ABI,
esta información permite a la librería Web3 conocer
qué funciones están implementadas en el Smart Contract*/
var PIController = web3.eth.contract("YOUR ABI");

//Se indica la dirección del Smart Contract con el que interactuar
var PI=  PIController.at('0x916E54853a6CAe90b918adE5496B63Df7fF0d5d5');

/*Funciones "Get" que leen el estado del Smart Contract
al recargar la interfaz web*/
PI.getReference(function(error, result){
    if(!error)
    {
        $("#Reference-read").html('Reference:'+result/100);
        console.log(result);
```

```
    }
    else
       console.error(error);
       });

PI.getKp(function(error, result){
      if(!error)
      {
          $("#Kp-read").html('Kp:'+result);
          console.log(result);
      }
      else
          console.error(error);
  });

PI.getKi(function(error, result){
    if(!error)
    {
        $("#Ki-read").html('Ki:'+result);
        console.log(result);
    }
    else
        console.error(error);
});

PI.getInputSignal(function(error, result){
  if(!error)
  {
      $("#InputSignal-read").html('Temperature:'+result/100);
      console.log(result);
  }
  else
      console.error(error);
      });

PI.getControlSignal(function(error, result){
  if(!error)
  {
      $("#ControlSignal-read").html('u=ControlSignal:'+result/100);
      console.log(result);
  }
  else
      console.error(error);
      });
PI.getError(function(error, result){
    if(!error)
    {
```

```
    $("#Error-read").html('Error=Reference-Temperature:'+result);
    console.log(result);
    }
    else
        console.error(error);
    });

PI.getError1(function(error, result){
  if(!error)
  {
  $("#Error1-read").html('ErrorAcumulado=error_1+error:'+result);
  console.log(result);
  }
  else
     console.error(error);
  });

/*Eventos que permiten monitorizar el Smart Contract sin recargar
continuamente la página o sin ejecutar continuamente funciones Get*/
var RefEvent=PI.RefEvent({},'latest');
var KpEvent=PI.KpEvent({},'latest');
var KiEvent=PI.KiEvent({},'latest');
var ControlSignalEvent=PI.ControlSignalEvent({},'latest');
var setError_1Event=PI.setError_1Event({},'latest');

RefEvent.watch(function(err, result){
if (!err) {
if (result.blockHash != $("#instrans").html())
    $("#loader").hide();
    $("#insTrans").html('Block hash: ' +result.blockHash);
    $("#Reference-read").html('Reference by event:
        '+result.args.Reference/100);
        }
else
{
    $("#loader").hide();
    console.log(err);
        }
});

KpEvent.watch(function(err, result){
if (!err) {
if (result.blockHash != $("#instrans").html())
          $("#loader").hide();
          $("#insTrans").html('Block hash: ' +result.blockHash);
          $("#Kp-read").html('Kp by event:'+result.args.Kp);
          }
```

```
else
{
      $("#loader").hide();
      console.log(err);
          }
});

KiEvent.watch(function(err, result){
if (!err) {
if (result.blockHash != $("#instrans").html())
        $("#loader").hide();
        $("#insTrans").html('Block hash: ' +result.blockHash);
        $("#Ki-read").html('Ki by event:'+result.args.Ki);
        }
else
{
    $("#loader").hide();
    console.log(err);
        }
});

ControlSignalEvent.watch(function(err, result){
if (!err) {
if (result.blockHash != $("#instrans").html())
$("#loader").hide();
$("#insTrans").html('Block hash: ' +result.blockHash);
$("#InputSignal-read").html('Temperature by event:'+result.args.X/100);
$("#ControlSignal-read").html('u=ControlSignal by event:
'+result.args.ControlSignalAdap/100);
$("#Error-read").html('Error=Reference-Temperature by event:
'+result.args.error);
$("#Error1-read").html('ErrorAcumulado=error_1+error by event:
'+result.args.error_1);
          }
else
{
  $("#loader").hide();
  console.log(err);
          }
});

setError_1Event.watch(function(err, result){
if (!err) {
if (result.blockHash != $("#instrans").html())
$("#loader").hide();
$("#insTrans").html('Block hash: ' +result.blockHash);
$("#Error1-read").html('ErrorAcumulado=error_1+error by event:
```

```
'+result.args.error_1);
        }
else
{
    $("#loader").hide();
    console.log(err);
        }
});

/*Botones web que permiten ejecutar funciones Set en el Smart Contract
al pulsar el botón se llama a Metamask para que firme la transacción
y envíe el parámetro correspondiente a la función invocada*/

$("#buttonRef").click(function() {
  $("#loader").show();
    PI.setReference($("#Reference").val(), (err, res) => {
if (err)
    $("#loader").hide();
    });
});

$("#buttonKp").click(function() {
  $("#loader").show();
    PI.setParameterKp($("#Kp").val(), (err, res) => {
if (err)
    $("#loader").hide();
    });
});
$("#buttonKi").click(function() {
  $("#loader").show();
    PI.setParameterKi($("#Ki").val(), (err, res) => {
if (err)
    $("#loader").hide();
    });
});
$("#buttonErrorAc").click(function() {
  $("#loader").show();
    PI.setError_1($("#Error_1").val(), (err, res) => {
if (err)
    $("#loader").hide();
    });
});
```

En el código HTML se diferencian 4 bloques:

1. Definición de comunicación con el Smart Contract
2. Petición de las variables al Smart Contract

3. Actualización de las variables mediante monitorización de eventos en la Blockchain

4. Utilización de botones para actualizar las variables en la Blockchain, al utilizar los botones se realiza una llamadas a la extensión Metamask que permite firmar transacciones desde una interfaz web

7.5 Despliegue

7.5.1 Despliegue del código Arduino

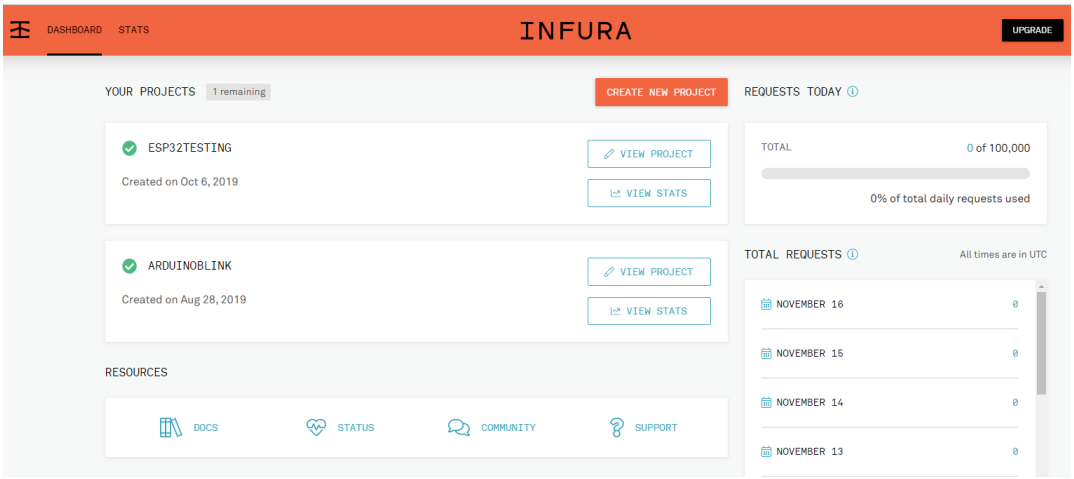
Para cargar el código desarrollado en Arduino se utiliza la interfaz Arduino IDE que permite fácilmente compilar y escribir en el Arduino ESP32 el código a través de un puerto serie virtual, simplemente usando una conexión USB, si se desea realizar cambios en el código es necesario conectar el Arduino de nuevo y sustituir el código antiguo por el nuevo.

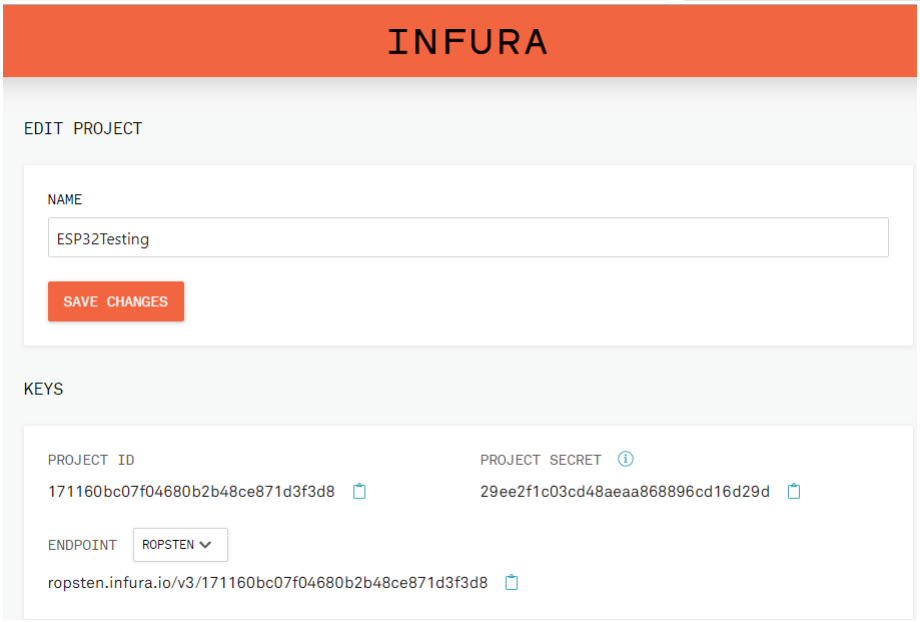
Configuración del Nodo Infura

Es necesario configurar un nodo Infura para que el Arduino pueda escribir en la Blockchain.

Como se observa en la figura 7.7, configurar un nodo en Infura es muy sencillo, una vez registrado, crear un nuevo proyecto, indicar la Blockchain y se genera automáticamente un "Endpoint" propio al que poder enviar transacciones, en nuestro caso desde el Arduino.

Infura también permite llevar un seguimiento de las transacciones que recibe el nodo que se ha configurado.


DASHBOARD
STATS
INFURA
UPGRADE
YOUR PROJECTS
1 remaining
CREATE NEW PROJECT
REQUESTS TODAY
ESP32TESTING
Created on Oct 6, 2019
VIEW PROJECT
VIEW STATS
TOTAL
0 of 100,000
0% of total daily requests used
ARDUINOBLINK
Created on Aug 28, 2019
VIEW PROJECT
VIEW STATS
TOTAL REQUESTS
All times are in UTC
RESOURCES
DOCS
STATUS
COMMUNITY
SUPPORT
NOVEMBER 16
NOVEMBER 15
NOVEMBER 14
NOVEMBER 13


INFURA
EDIT PROJECT
NAME
ESP32Testing
SAVE CHANGES
KEYS
PROJECT ID
171160bc07f04680b2b48ce871d3f3d8
PROJECT SECRET
29ee2f1c03cd48aeaa868896cd16d29d
ENDPOINT
ROPSTEN
ropsten.infura.io/v3/171160bc07f04680b2b48ce871d3f3d8

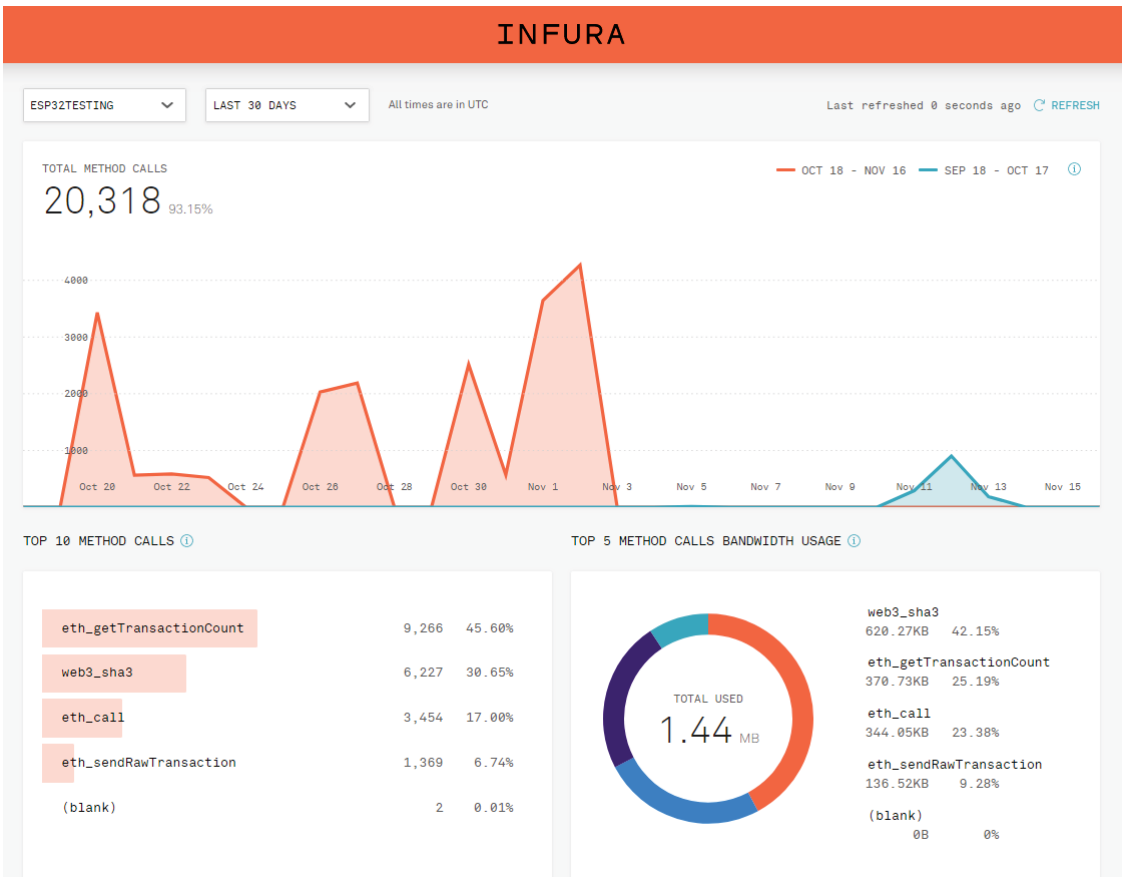


Figura 7.7 Configuración y monitorización del nodo en Infura.

7.5.2 Despliegue del código Solidity

El código Solidity para el Smart Contract se ha desarrollado en el editor de código **Atom**, que permite una navegación ágil entre los archivos de un proyecto, para montar y desplegar el proyecto Solidity se ha utilizado la herramienta Truffle, que realiza las siguientes acciones:

1. Instala fácilmente un entorno de desarrollo de Smart Contract, donde solo es necesario por parte del usuario empezar a desarrollar el Smart Contract con Atom en la carpeta correspondiente y configurar la Blockchain a la que conectarse.
2. Incluye un entorno local de desarrollo y pruebas.
3. Permite compilar un Smart Contract en Solidity.
4. Permite desplegar un Smart Contract en la Blockchain de Ethereum (mainnet y/o testnet).

Los pasos para desarrollar un Smart Contract:

1. Ejecutar ***"Truffle init"*** en la carpeta deseada, esto crea la estructura del proyecto necesaria.
2. Desarrollar el Smart Contract con Atom en la carpeta ***"Contracts"***.

3. Compilar con ***"Truffle compile"***.

4. Desplegar el Smart Contract con ***"Truffle Deploy"*** en un nodo local (para pruebas) o en la Blockchain de Ethereum (mainnet y/o testnet) a través de un nodo (local o remoto), en nuestro caso se utiliza un nodo en **Infura.io**, en la figura 7.8 se muestra el despliegue del Smart Contract PIController.

5. Comprobar que en la Blockchain de Ropsten se ha creado el Smart Contract, en la figura 7.9 se observa que se ha creado correctamente el Smart Contract en Ropsten.

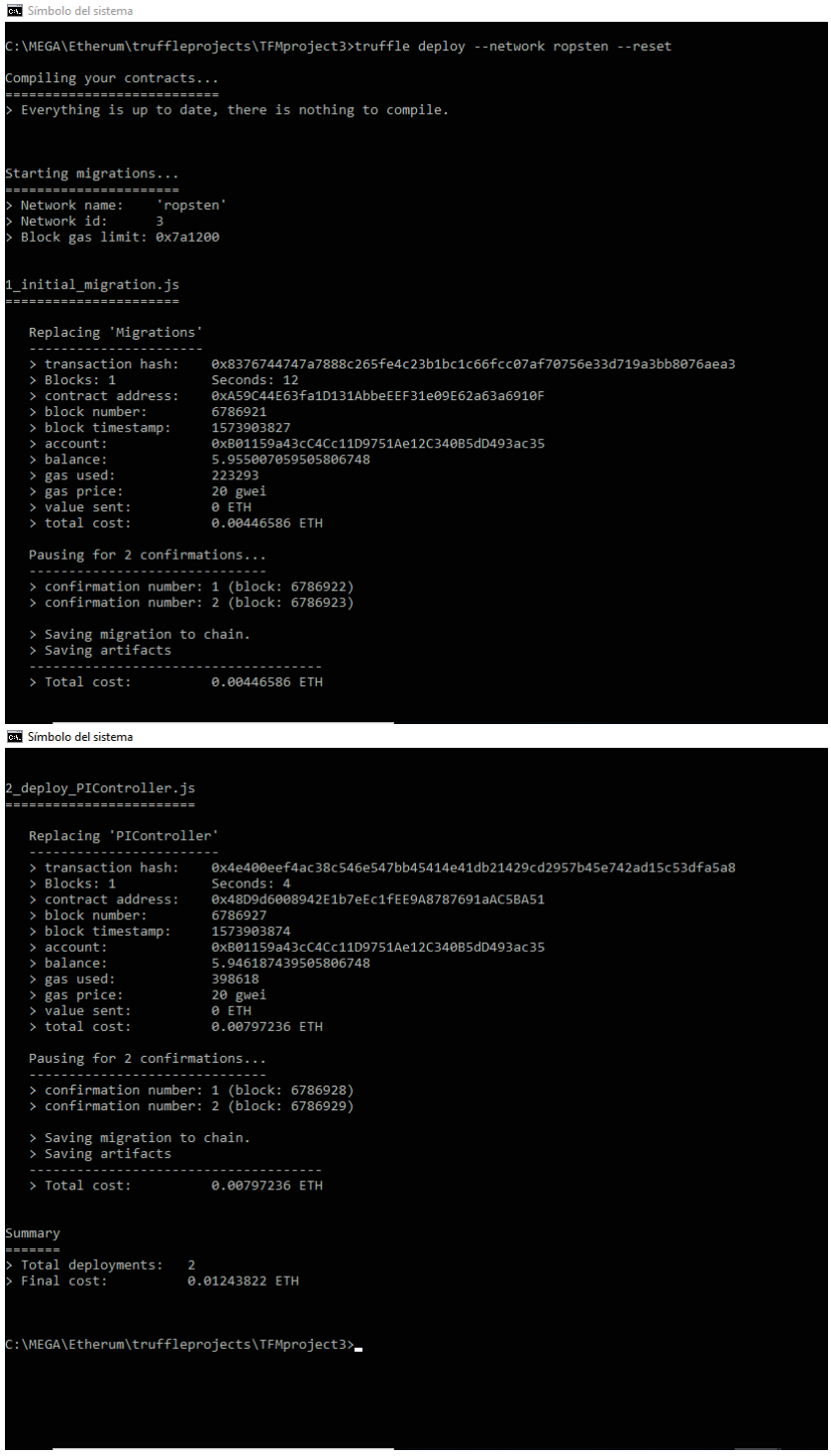


Figura 7.8 Despliegue del Smart Contract en Ropsten.

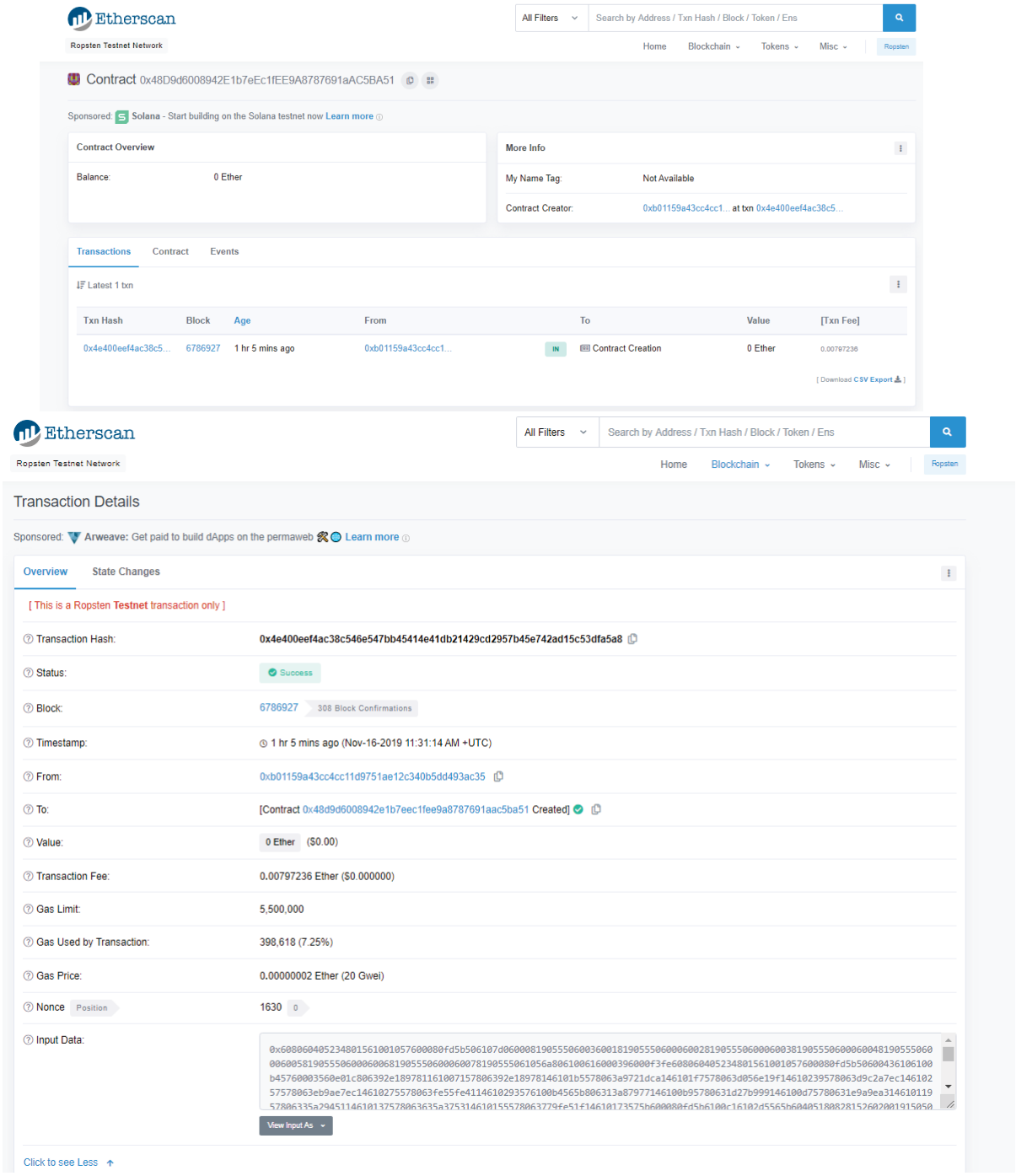


Figura 7.9 Despliegue del Smart Contract en Ropsten, transacción en la Blockchain.

7.5.3 Despliegue del código HTML

Se ha utilizado **Node.js**, que permite desarrollar código HTML con librerías JavaScript para interactuar con Blockchain, y permite crear un servidor local en el que ejecutar el código web desarrollado, ver figura 7.10.

Este servidor local además comunica con la extensión web de **Metamask** que permite firmar transacciones y enviarlas a la Blockchain conectándose a los nodos de Metamask, en la figura 7.11 se observa cómo se manda un cambio de referencia, se dispara la extensión de Metamask para firmar la transacción y como la interfaz web actualiza el valor de

la referencia por evento cuando el Smart Contract en la Blockchain valida la transacción.

Por último en la figura 7.12 se muestra como llega a la Blockchain y queda recogido el cambio del valor de la Referencia, que es inmutable y que puede consultarse en cualquier momento desde cualquier lugar del mundo.

Los pasos a seguir son:

1. Descargar e instalar Node.js y las librerías Web3.
2. Desarrollar código HTML utilizando en nuestro caso Atom.
3. Configurar y ejecutar un servidor web local para interactuar con el código web desarrollado.

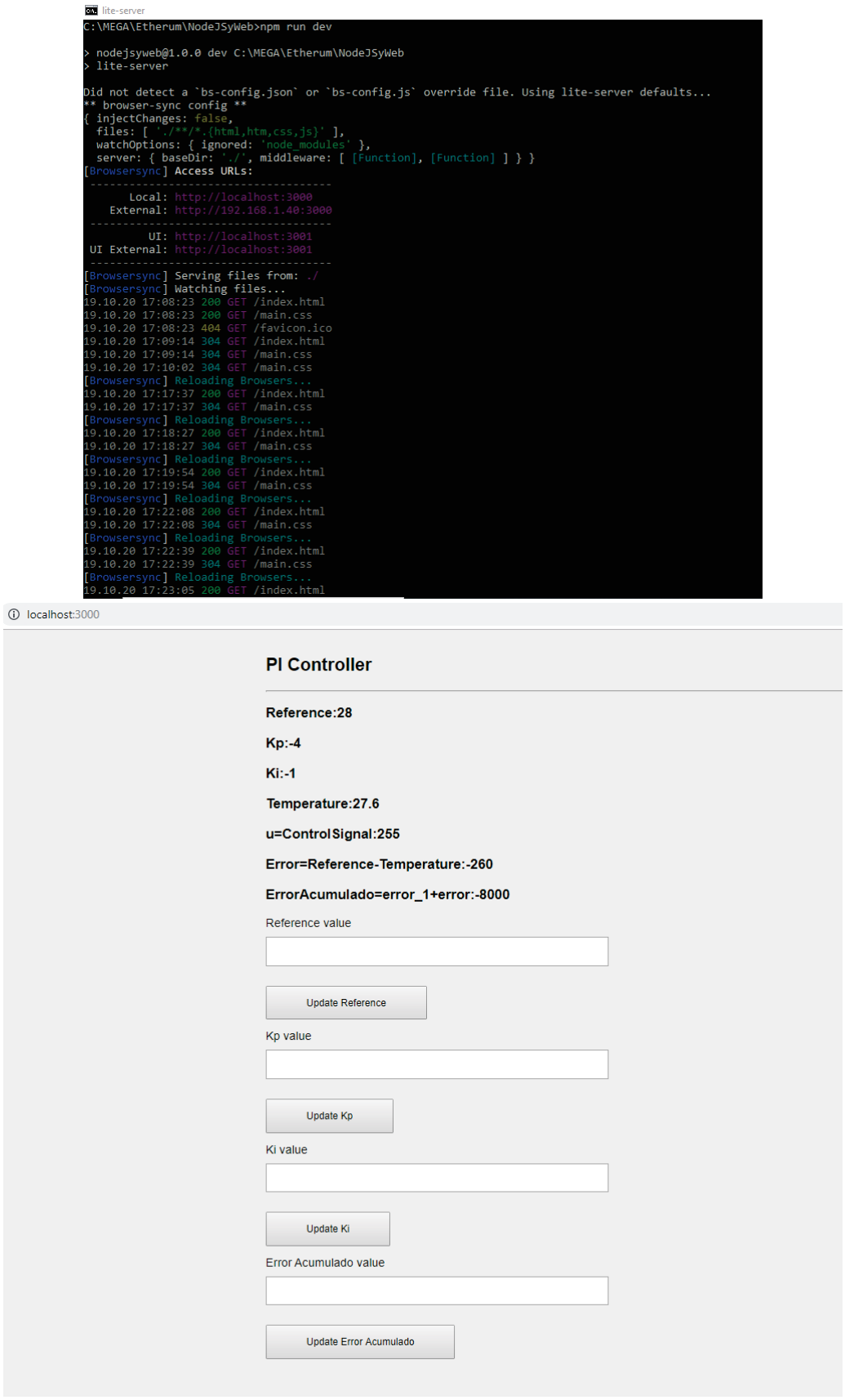


Figura 7.10 Funcionamiento del webserver e interfaz web.

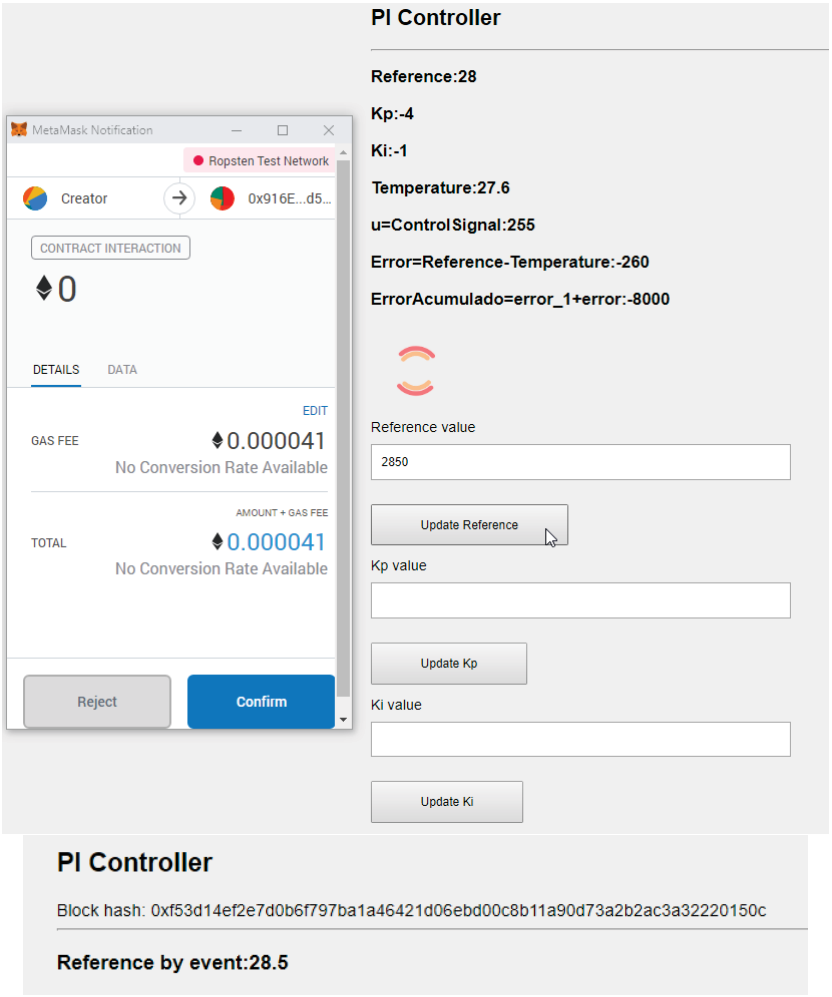


Figura 7.11 Funcionamiento de interfaz web y Metamask.

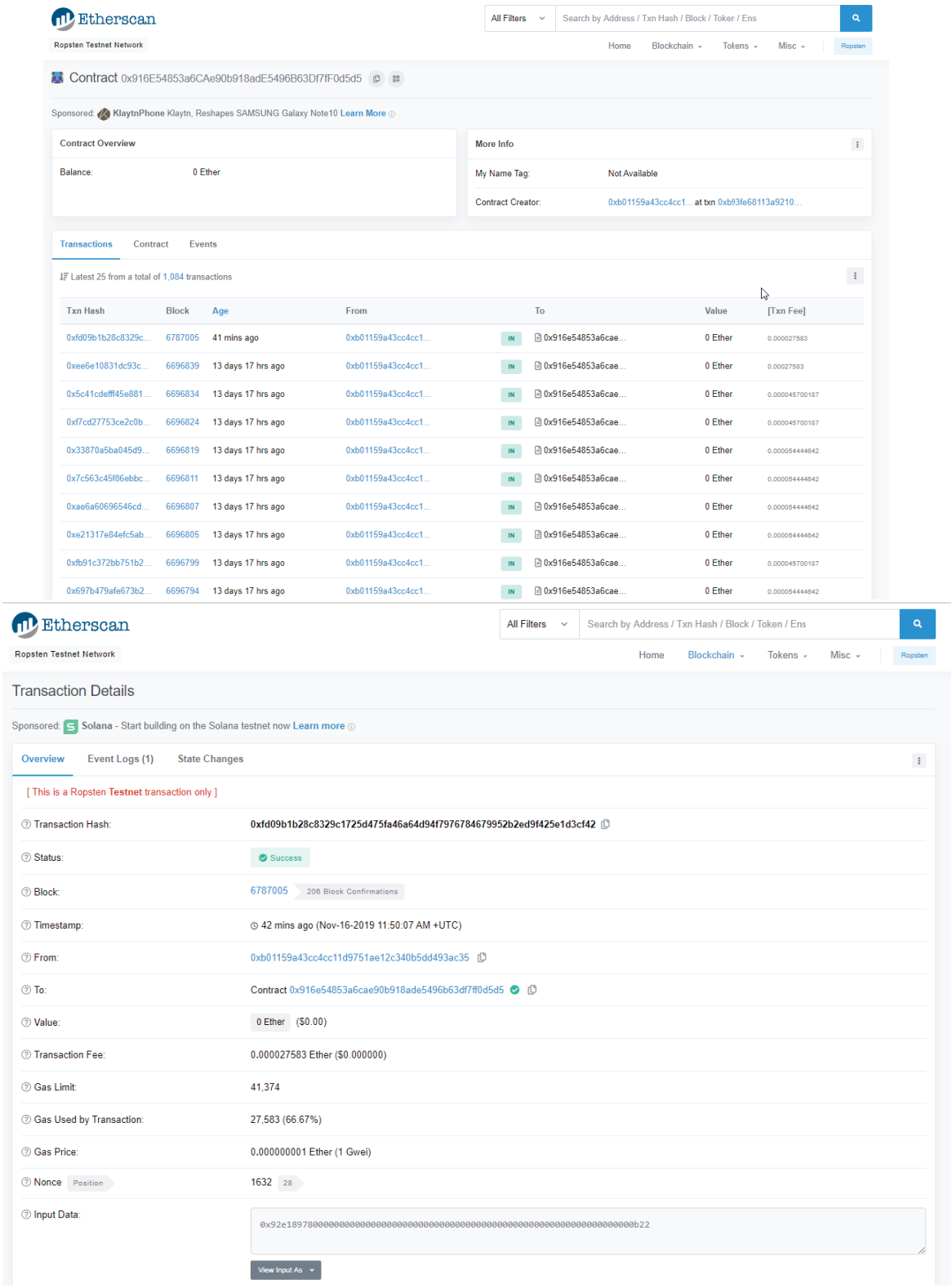


Figura 7.12 Transacciones recibidas y recogidas en el Smart Contract.

7.6 Experimentación

7.6.1 Diagrama conexiones del proyecto

En la figura 7.13 se observa el diagrama completo de conexiones del proyecto, es importante hacer notar que el Arduino escribe y lee del Smart Contract (que este último se ejecuta de manera independiente en la Blockchain) utilizando un nodo de **Infura**, mientras en **paralelo** la interfaz web escribe y lee del mismo Smart Contract utilizando un nodo de **Metamask**, no siendo necesario que la interfaz web y el Arduino estén en la **misma red**, ni tan siquiera el mismo país, solo es necesario tener acceso a **Internet** para poder conectarse a un nodo P2P de la Blockchain de Ethereum Ropsten.

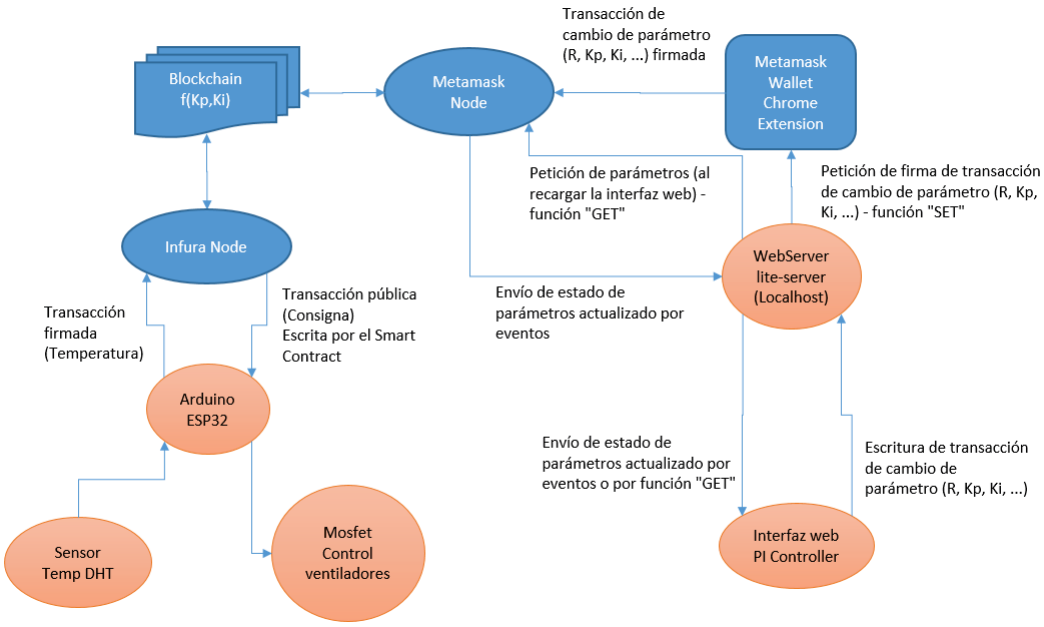


Figura 7.13 Diagrama completo de conexiones (Elaboración propia, 2019).

7.6.2 Implementación

Una vez desplegados y probados todos los códigos y aplicaciones, comienza la parte de experimentación.

1. Se coloca el sensor de temperatura con los ventiladores enfrente de la salida de aire caliente de un portátil para usarlo como fuente de calor y de perturbaciones.

2. Se alimenta el Arduino y este empieza a buscar la red Wi-Fi indicada.

3. Se conecta a la red Wi-Fi y se procede a la lectura de la temperatura, una vez leída se establece una referencia cercana al punto de funcionamiento a través de la interfaz web.

4. Comienza el control de los ventiladores, cuando la temperatura del sensor se ha establecido en torno al punto de funcionamiento se procede a cambiar los Setpoints para realizar varios experimentos de sintonizado del PI variando los parámetros de Kp y Ki del Smart Contract a través de la interfaz web.

5. Una vez sintonizado el PI se da comienzo a realizar varias pruebas de control de subida y bajada de Setpoint.

7.6.3 Resultados

Control en torno a punto de funcionamiento

Se observa en la figura 7.14 que tras el arranque del controlador PI la señal de control PWM generada va en aumento (aceleración de los ventiladores) para hacer frente al aumento de temperatura en el sensor, la temperatura comienza a **bajar** hasta alcanzar el Setpoint, a partir de ese momento la señal de control va variando levemente para que la temperatura leída en el sensor oscile ligeramente(+-0.1ºC) alrededor del setpoint.

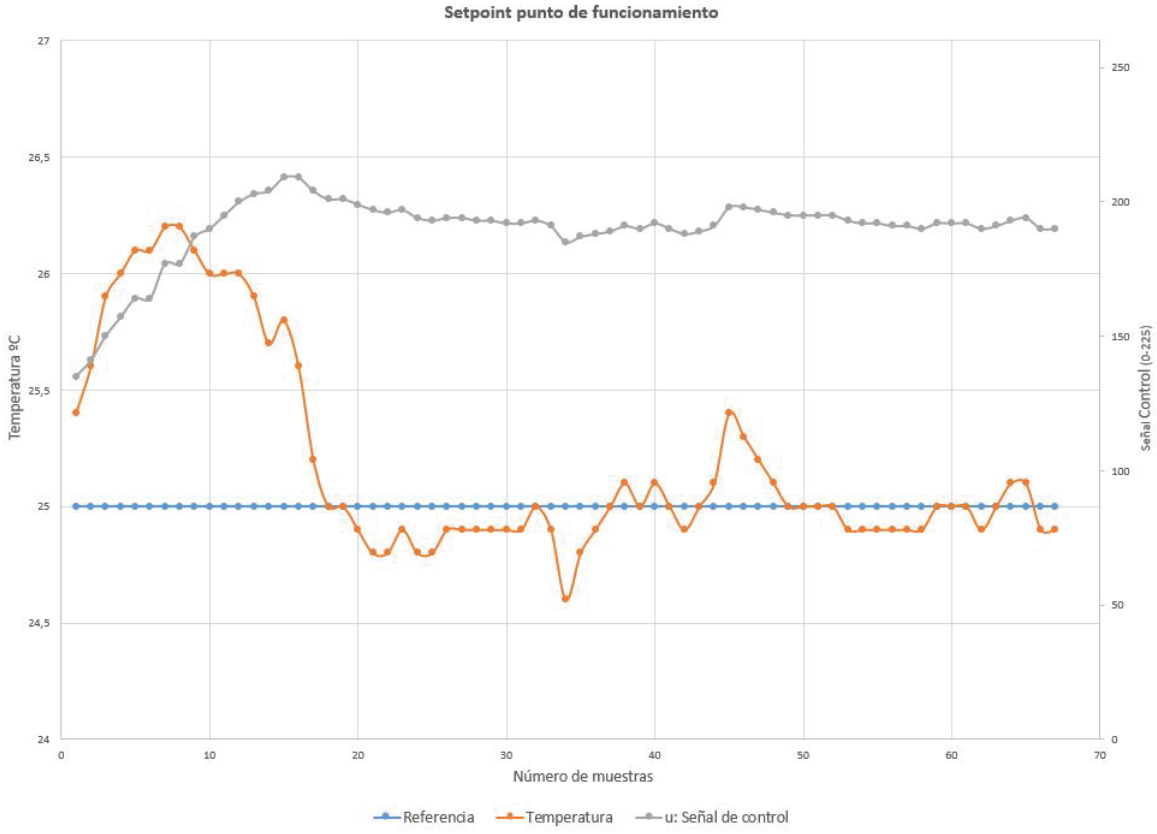


Figura 7.14 Gráfica: Control en torno a punto de funcionamiento.

Se observa que el sistema tiene una inercia al cambio de temperatura alta al inicio, debido a que la fuente de calor continúa aportando energía y los ventiladores no tienen capacidad de disipar todo el calor, pero en cuanto comienzan a pasar los ciclos el efecto del diferencial de temperatura y del término integral se hace notar conduciendo al sistema al punto de funcionamiento deseado.

Se obtiene la tabla de resultados 7.2:

Parámetro	Valor	Comentarios
Temperatura inicial	25.4ºC	Temperatura en aumento
Setpoint inicial	25ºC	
Setpoint final	25ºC	Punto de funcionamiento
Tiempo de establecimiento	18 ciclos	
Sobreoscilación $(100 \cdot (T_{pico} - T_\infty)/T_\infty)$	-0.8 %	
Error Relativo $(100 \cdot (R - T_\infty)/R)$	0-0.4 %	En régimen permanente oscila en torno a 0.1ºC
Estabilidad	Alta	
Saturación	No	En este experimento no satura la señal de control

Tabla 7.2 Resultados: Control en torno a punto de funcionamiento.

Control con setpoint de bajada

Los siguientes experimentos corresponder al envío de consignas de bajar temperatura.

Se observa en la figura 7.15 que tras el envío del nuevo setpoint al controlador PI la señal de control PWM generada va en aumento logrando bajar la temperatura hasta alcanzar el Setpoint de 25.5ºC.

Una vez alcanzado el Setpoint se observa que los ventiladores no son capaces de mantener la temperatura debido a un aumento de temperatura de la fuente de generación de calor, por ello el controlador PI sigue aumentando la señal de control hasta llegar al valor de 255 que equivale al 100 % del PWM, cuando satura la señal de control los ventiladores están a su máxima capacidad y el sistema de estabiliza en torno a 27-27.1ºC.

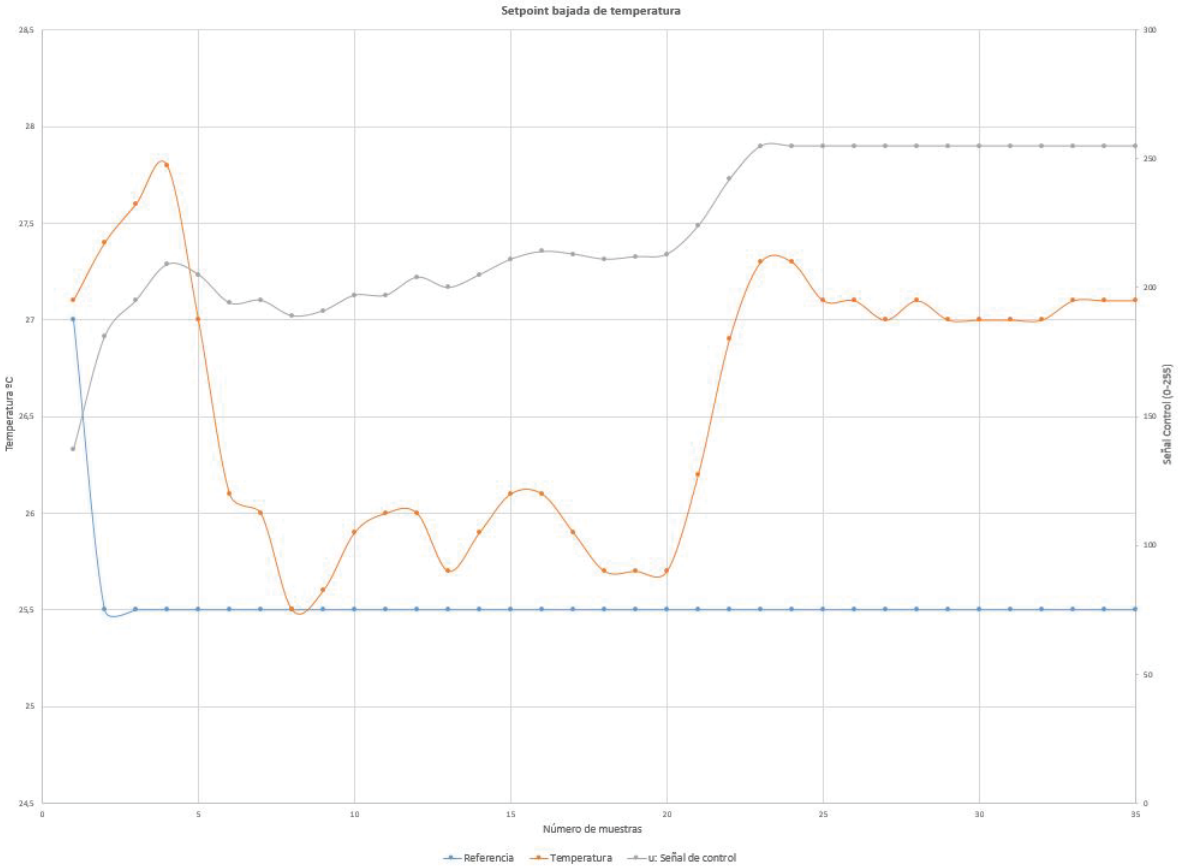


Figura 7.15 Gráfica: Control con Setpoint de bajada, test 1.

Parámetro	Valor	Comentarios
Temperatura inicial	27.1ºC	Temperatura en aumento
Setpoint inicial	27ºC	
Setpoint final	25.5ºC	
Tiempo de establecimiento	29 ciclos	
Sobreoscilación $(100 \cdot (T_{pico} - T_\infty)/T_\infty)$	0 %	
Error Relativo $(100 \cdot (R - T_\infty)/R)$	6.27 %	La señal de control satura y no es posible alcanzar el setpoint
Estabilidad	Media	El sistema se estabiliza pero no alcanza el Setpoint
Saturación	Si	En este experimento satura la señal de control

Tabla 7.4 Resultados: Control con setpoint de bajada, test 1.

Se realiza otro experimento de bajada de setpoint, como se observa en la figura 7.16, tras el cambio de setpoint la señal de control PWM generada va en aumento logrando bajar

la temperatura hasta alcanzar la temperatura de 27.6ºC, una vez alcanzada esa temperatura se observa que los ventiladores no son capaces de alcanzar la temperatura deseada, cuando satura la señal de control el sistema de estabiliza en torno a 27.6ºC.

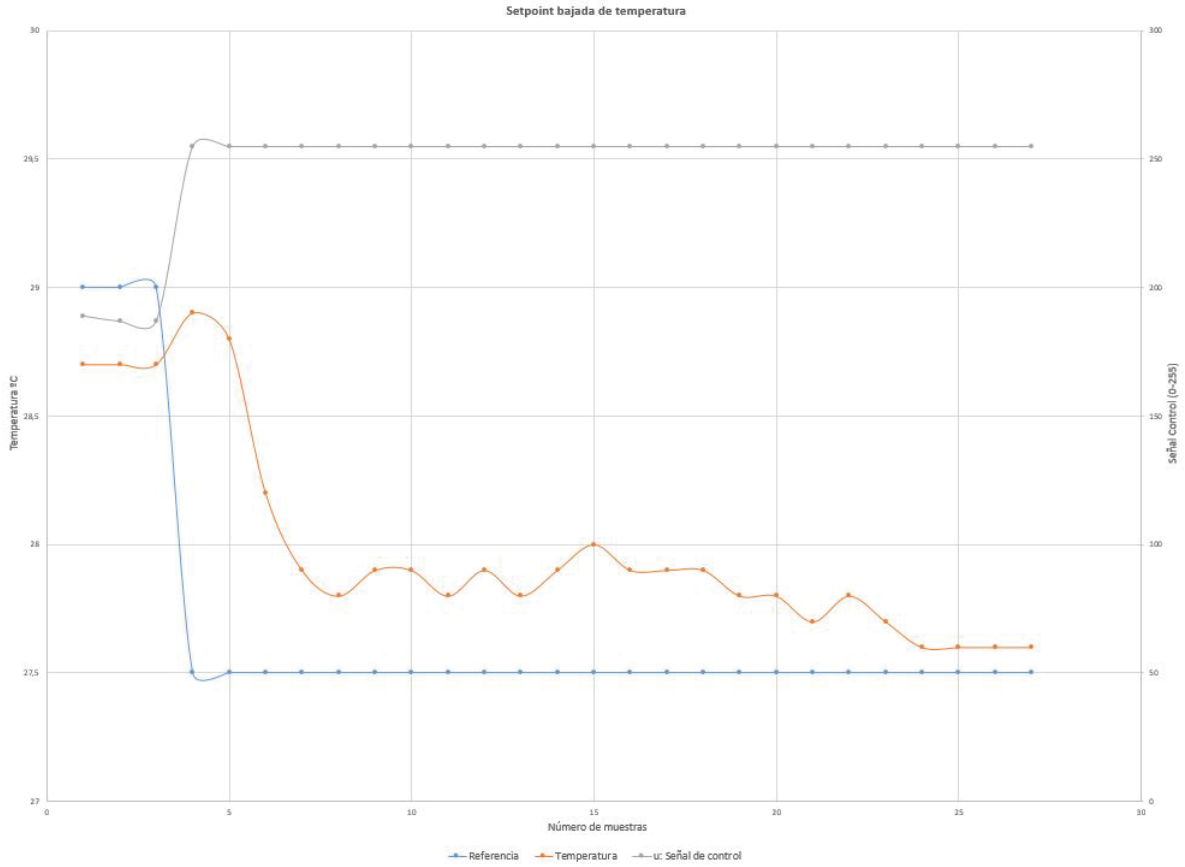


Figura 7.16 Gráfica: Control con setpoint de bajada, test 2.

Parámetro	Valor	Comentarios
Temperatura inicial	28.7ºC	Temperatura en aumento
Setpoint inicial	29ºC	
Setpoint final	27.5ºC	
Tiempo de establecimiento	25 ciclos	
Sobreoscilación $(100 \cdot (T_{pico} - T_\infty)/T_\infty)$	0 %	
Error Relativo $(100 \cdot (R - T_\infty)/R)$	0.36 %	La señal de control satura y no es posible alcanzar el setpoint
Estabilidad	Media	El sistema se estabiliza pero no alcanza el Setpoint
Saturación	Si	En este experimento satura la señal de control

Tabla 7.6 Resultados: Control con setpoint de bajada, test 2.

Se realiza otro experimento de bajada de setpoint, como se observa en la figura 7.17, tras el cambio de setpoint la señal de control PWM generada va en aumento logrando bajar la temperatura hasta alcanzar la temperatura de 27.1ºC, se produce una sobreoscilación, y se observa como la señal de control disminuye para mitigar esa sobreoscilación y finalmente alcanzando el setpoint objetivo.

Una vez alcanzado el setpoint se observa en la muestra número 34 una perturbación en la fuente de calor con una bajada de temperatura, en los siguientes ciclos el controlador disminuye la señal de control a los ventiladores para aumentar la temperatura y alcanzar el setpoint, en este experimento no hay saturación de la señal de control.

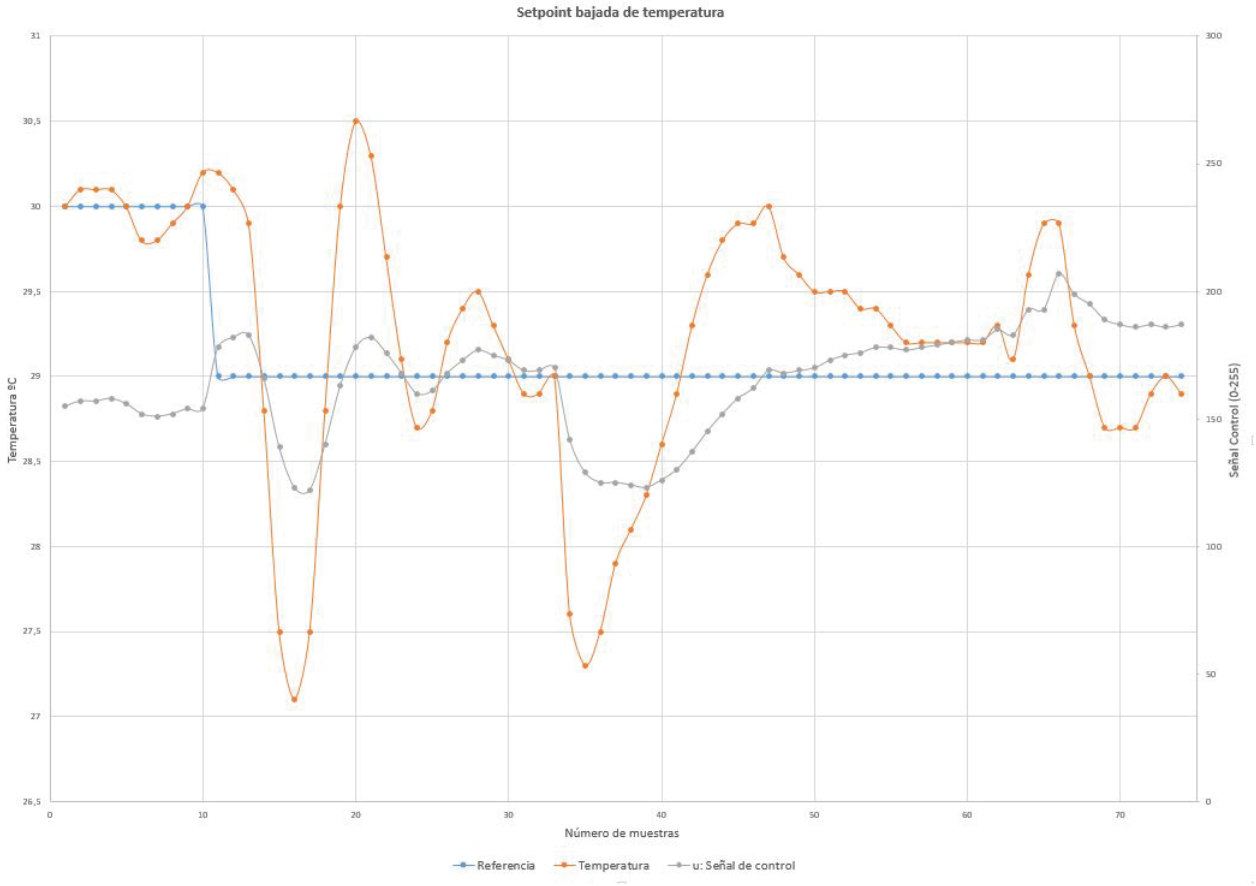


Figura 7.17 Gráfica: Control con setpoint de bajada, test 3.

Parámetro	Valor	Comentarios
Temperatura inicial	30ºC	Temperatura en aumento
Setpoint inicial	30ºC	
Setpoint final	29ºC	
Tiempo de establecimiento	34 ciclos y 40 ciclos tras perturbación	En total 74 ciclos para establecerse
Sobreoscilación $(100 \cdot (T_{pico} - T_\infty)/T_\infty)$	-6.55 %	
Error Relativo $(100 \cdot (R - T_\infty)/R)$	0-0.4 %	En régimen permanente oscila en torno a 0.1ºC
Estabilidad	Media	Se estabiliza pero en este experimento sufre perturbaciones
Saturación	No	

Tabla 7.8 Resultados: Control con setpoint de bajada, test 3.

Control con setpoint de subida

Para este experimento se han realizado 3 setpoints de subida de temperatura.

Se observa en la figura 7.18 que el experimiento comienza con el controlador saturado para intentar alcanzar un setpoint de 25.5ºC, pero la temperatura oscila en torno a 27.1ºC.

- El primer setpoint que se manda es de subida de temperatura a 28ºC, se observa como el controlador PI deja de saturar y genera una señal de control que envía a los ventiladores al 50 % de potencia, esto provoca una subida brusca en la temperatura y el controlador actúa correctamente aumentando la señal de control y permitiendo alcanzar el setpoint de temperatura deseado disminuyendo la sobreoscilación.

- El segundo setpoint que se aplica, en comparación con el primero, el sistema no estaba en una situación de saturación y el diferencial de setpoint es menor, se consigue un control más suave, donde el controlador disminuye más suavemente la potencia de los ventiladores y la temperatura aumenta también suavemente, sin sobreoscilación.
  Al final del periodo observado (muestra 72), se detecta una perturbación en el aumento de temperatura y se observa como el controlador comienza a corregir levemente la temperatura.

- El tercer setpoint, ocurre en condiciones similares al segundo setpoint, y se observa como el controlador disminuye la señal de control y en este caso se alcanza el setpoint más rápidamente y con una leve sobreoscilación.

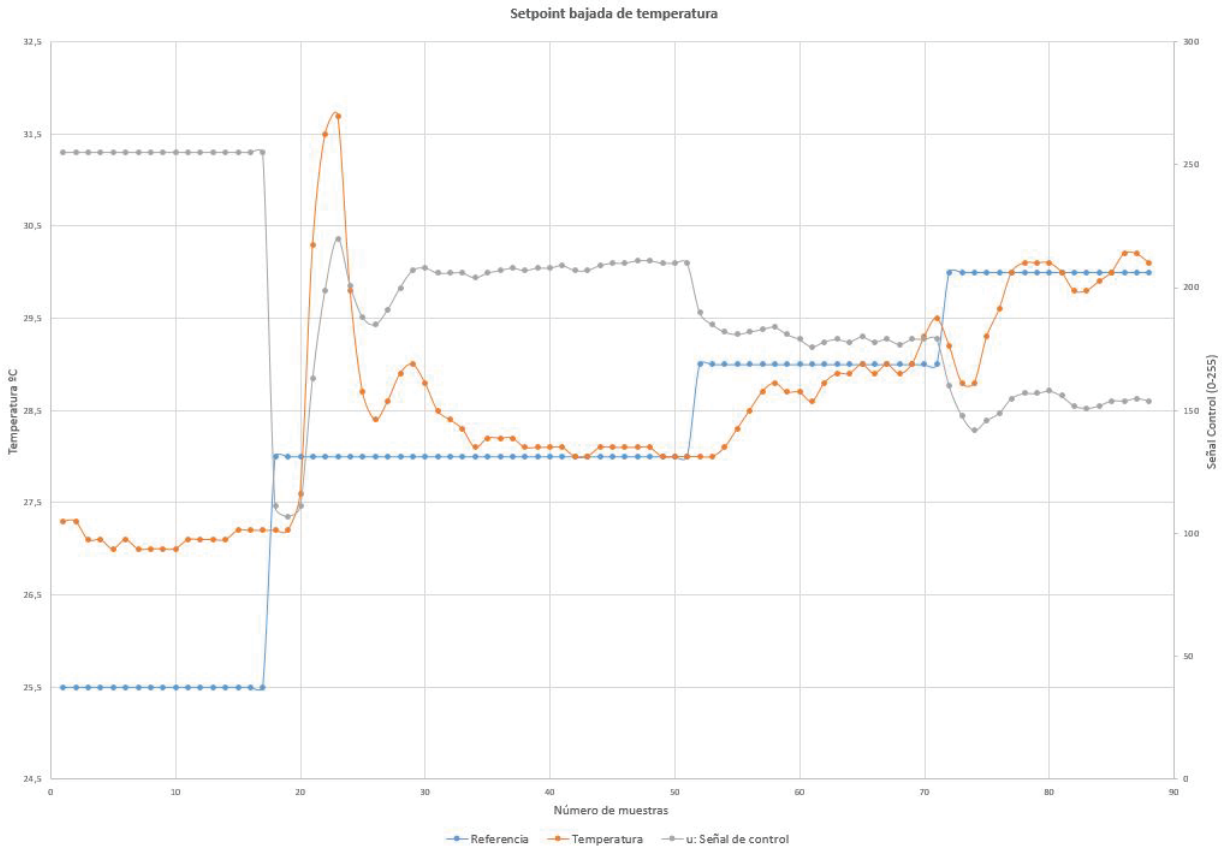


Figura 7.18 Gráfica: Control con Setpoint de subida.

Parámetro	Valor	Comentarios
Temperatura inicial	27.2ºC	Temperatura en aumento
Setpoint inicial	25.5ºC	
Setpoint final	28ºC	
Tiempo de establecimiento	21 ciclos	
Sobreoscilación $(100 \cdot (T_{pico} - T_\infty)/T_\infty)$	13.2 %	
Error Relativo $(100 \cdot (R - T_\infty)/R)$	0 %	
Estabilidad	Alta	
Saturación	No	

Tabla 7.10 Resultados: Control con setpoint de subida, test 1.

Parámetro	Valor	Comentarios
Temperatura inicial	28ºC	Temperatura en aumento
Setpoint inicial	28ºC	
Setpoint final	29ºC	
Tiempo de establecimiento	20 ciclos	
Sobreoscilación $(100 \cdot (T_{pico} - T_\infty)/T_\infty)$	0 %	
Error Relativo $(100 \cdot (R - T_\infty)/R)$	0 %	
Estabilidad	Media	Se estabiliza pero en este experimento sufre perturbaciones
Saturación	No	

Tabla 7.12 Resultados: Control con setpoint de subida, test 2.

Parámetro	Valor	Comentarios
Temperatura inicial	29.2ºC	Temperatura en aumento
Setpoint inicial	29ºC	
Setpoint final	30ºC	
Tiempo de establecimiento	18 ciclos	
Sobreoscilación $(100 \cdot (T_{pico} - T_\infty)/T_\infty)$	0.33 %	
Error Relativo $(100 \cdot (R - T_\infty)/R)$	0-0.4 %	Oscila en torno a 0.1ºC
Estabilidad	Media	
Saturación	No	

Tabla 7.14 Resultados: Control con setpoint de subida, test 3.

7.6.4 Ventajas e inconvenientes

Tras la realización de los experimentos se pueden extraer las siguientes ventajas e inconvenientes del uso de Blockchain en el control automático.

Ventajas:

1. Se logra una red más robusta a ataques y fallos
2. Las transacciones quedan registradas y son inmutables
3. Consigues una red más fiable
4. Se pueden obtener las propiedades nombradas anteriormente a bajo coste, no es necesario un hardware de control caro
5. Es una tecnología transparente, pero puede ser tan privada como se desee
6. En este caso particular se elimina el controlador central como posible punto de fallo
7. Se puede acceder e interactuar con el Smart Contract simplemente teniendo conexión a la red
8. Los resultados del Smart Contract son deterministas y la calidad de los datos es alta, hay seguridad de dato y evita su duplicidad

Inconvenientes:

1. Es una tecnología disruptiva que todavía es difícil de implementar
2. Todavía en fase experimental en cuanto a mecanismos de Consenso
3. Los datos no se pueden modificar ni por el creador
4. La gestión de las claves privadas es aún un reto

5. La Blockchain es más ineficiente que un sistema local de control, para el experimento se ha usado la Testnet Ropsten de Ethereum que trabaja con unos tiempos de entre 10-20 segundos por Bloque minado, por tanto 10-20 segundos por transacción validada en la Blockchain

6. Es de bajo coste pero no es gratuita, se ha usado la Testnet Ropsten que sí es gratuita, pero la Mainnet de Ethereum requiere el pago de "GAS" con su criptomoneda Ethereum como alquiler del poder computacional de sus mineros, es un pago muy bajo, pero si es un proceso muy reiterativo puede ser un gran coste.
   Para evitar esta situación, la solución Blockchain al problema que se presente puede desarrollarse en una Blockchain privada (Ethereum es pública) o utilizar otro proyecto Blockchain como Hyperledger que es una iniciativa para crear aplicaciones Blockchain industriales sin criptomonedas

7. Almacenamiento puede ser un inconveniente para nodos completos pequeños

8 Conclusiones

Este proyecto presenta un ejemplo de automatización dentro del ámbito domótico. En concreto la climatización de edificios. Este proceso de automatización permite controlar dispositivos mediante el envío de consignas utilizando Blockchain y Smart Contracts como tecnología conductora del bucle de control automático y las comunicaciones.

Al realizar este trabajo se concluye que la tecnología Blockchain tiene aplicación en campos distintos a las criptomonedas, se comprueba que es posible realizar un controlador PI para dispositivos IoT utilizando esta tecnología y confiriendo al controlador y a la red IoT las propiedades de verdad, confianza y seguridad de la Blockchain.

Con los experimentos realizados se demuestra que es posible controlar dispositivos conectados a Internet con mayor seguridad, más certeza, sin aumento significativo de coste, se pretende dar una perspectiva nueva de esta tecnología, aplicable a al campo del control automático, y abre la puerta a la aplicación en más campos relacionados con la informática, redes, las relaciones en la red, la confianza y la seguridad.

Desde el punto de vista del programador, los lenguajes de programación y la tecnología aún tienen un gran camino que recorrer, pero se demuestra que ya es posible conectar dispositivos IoT con Blockchain. Es un Software que requiere tiempo y práctica antes de poder ser usado de forma fluida. A parte de los conocimientos en Solidity, ha sido necesario adquirir conocimientos profundos en Arduino y una introducción a HTML con Node.js.

Se ha mostrado en el documento que Blockchain es una tecnología joven (10 años) y aún en fase de lanzamiento tecnológico pero con gran potencial, que puede revolucionar la forma en la que interactúa el ser humano con la red actualmente, permitiendo transformar la presente dependencia de las Third Parties.

En relación a lo antes expuesto, podemos deducir que existe un gran campo de investigación en el proceso de integrar Blockchain al mundo actual y que pase a estar integrado en la sociedad como ya ha logrado establecerse Internet.

8.1 Líneas futuras

Existen diversas opciones para continuar en el campo que se inicia en este Proyecto, sin embargo, se presentan unas ideas a continuación.

Como extensión, se sugiere integrar más dispositivos en el Smart Contract para testear las capacidades de este para múltiples bucles de control en paralelo.

Sustituir el nodo infura por nodos ligeros ejecutándose en los propios dispositivos IoT para aumentar la seguridad.

Desarrollar nuevos métodos de control como el control MPC (Model Based Predictive Control) o Control difuso en Smart Contract.

Referencias

99bitcoins (2019). *What is the Bitcoin Mempool?* Accedido el 08-07-2019. URL: *https://99bitcoins.com/bitcoin-mempool/*.

Academy, Lisk (2019). *Delegated Proof of Stake*. Accedido el 20-07-2019. URL: *https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/delegated-proof-of-stake*.

Aldakin (s.f.). *Automatización Industrial y robótica. Qué es y sus claves de éxito*. Accedido el 28-07-2019. URL: *http://www.aldakin.com/automatizacion-industrial-robotica-claves-exito/*.

Alon Gal (2018). *The Tangle an illustrated introduction. Part 5 Consensus, confirmation confidence, and the coordinator*. Accedido el 21-06-2019. URL: *https://blog.iota.org/the-tangle-an-illustrated-introduction-79f537b0a455*.

Álvarez, Raúl (2018). *La información privada de 30 millones de usuarios en riesgo tras el más reciente hackeo a Facebook*. Accedido el 15-06-2019. URL: *https://www.xataka.com/seguridad/informacion-privada-30-millones-usuarios-riesgo-reciente-hackeo-a-facebook-fbi-investigara-caso*.

Anders Brownworth (2016). *Blockchain 101 - Una demostración visual*. Accedido el 22-06-2019. URL: *https://www.youtube.com/watch?v=_160oMzblY8*.

Andreas M. Antonopoulos, aantonop (2014). *Andreas Antonopoulos on Bitcoin Wallet Encryption*. Accedido el 24-06-2019. URL: *https://www.youtube.com/watch?v=PdGRmshPXdo*.

— (2016). *Consensus Algorithms, Blockchain Technology and Bitcoin UCL - by Andreas M. Antonopoulos*. Accedido el 22-06-2019. URL: *https://www.youtube.com/watch?v=fw3WkySh_Ho&t=784s*.

Azis (2016). *Guía de Consenso Algoritmos: ¿Cuál es Mecanismo de Consenso?* Accedido el 21-06-2019. URL: *https://masterthecrypto.com/guide-to-consensus-algorithms-what-is-consensus-mechanism/?lang=es*.

Back, Adam (2002). *Hashcash - A Denial of Service Counter-Measure*. Accedido el 08-06-2019. URL: *http://www.hashcash.org/papers/hashcash.pdf*.

Bárbara Nogales (2019). *El próximo halving de Bitcoin*. Accedido el 21-06-2019. URL: *https://bitcoin.es/criptomonedas/el-proximo-halving-de-bitcoin/*.

Bayer, Dave, Stuart Haber y W Scott Stornetta (1993). «Improving the efficiency and reliability of digital time-stamping». En: *Sequences Ii*. Springer, págs. 329-334.

Bill Buchanan OBE (2018). *Cryptography: Public Key Encryption (RSA, Elliptic Curve and ElGamal)*. Accedido el 23-06-2019. URL: *https://www.youtube.com/watch?v=QEYqkxuzoTg*.

binance.vision (2018). *¿Qué son los nodos?* Accedido el 29-06-2019. URL: *https://www.binance.vision/es/blockchain/what-are-nodes*.

Bit2Me (2018a). *¿Qué es la Cadena de Bloques (Blockchain)?* Accedido el 16-06-2019. URL: *https://academy.bit2me.com/que-es-cadena-de-bloques-blockchain/*.

— (2018b). *Smart Contracts: ¿Qué son, cómo funcionan y qué aportan?* Accedido el 20-07-2019. URL: *https://academy.bit2me.com/que-son-los-smart-contracts/*.

— (s.f.[a]). *¿Qué es la clave pública?* Accedido el 24-06-2019. URL: *https://academy.bit2me.com/que-es-clave-publica/*.

— (s.f.[b]). *Qué es un bloque dentro de la blockchain*. Accedido el 30-06-2019. URL: *https://academy.bit2me.com/que-es-un-bloque-dentro-de-la-blockchain/*.

— (s.f.[c]). *Transacciones Bitcoin, ¿Cómo funcionan?* Accedido el 30-06-2019. URL: *https://academy.bit2me.com/transacciones-bitcoin/*.

bitcoin.it (2011a). *Bloque*. Accedido el 30-06-2019. URL: *https://es.bitcoin.it/wiki/Bloque*.

— (2011b). *Introduction: Provide a general overview of the Bitcoin system and economy*. Accedido el 08-06-2019. URL: *https://en.bitcoin.it/wiki/Help:Introduction*.

— (2011c). *Introduction: Sending payments*. Accedido el 15-06-2019. URL: *https://en.bitcoin.it/wiki/Help:Introduction#Sending_payments*.

— (2017). *Wallet import format*. Accedido el 24-06-2019. URL: *https://en.bitcoin.it/wiki/Wallet_import_format*.

— (2019). *Secp256k1*. Accedido el 23-06-2019. URL: *https://en.bitcoin.it/wiki/Secp256k1*.

Bitcoin.org (2019). *DPoS*. Accedido el 20-07-2019. URL: *https://en.bitcoinwiki.org/wiki/DPoS*.

bitcoin.org (2018). «Transactions». En:

bitcoinwiki.org (2019a). *Dificultad del minado*. Accedido el 09-07-2019. URL: *https://es.bitcoinwiki.org/wiki/Dificultad_del_minado*.

— (2019b). *Nodo*. Accedido el 29-06-2019. URL: *https://es.bitcoinwiki.org/wiki/Nodo*.

blockchainhub (2019). *Decentralized Applications, DApps*. Accedido el 21-07-2019. URL: *https://blockchainhub.net/decentralized-applications-dapps/*.

Bryant, Martin (2011). *20 years ago today, the World Wide Web opened to the public*. Accedido el 08-06-2019. URL: *https://thenextweb.com/insider/2011/08/06/20-years-ago-today-the-world-wide-web-opened-to-the-public/*.

Buterin, Vitalik (2017). *The Meaning of Decentralization*. Accedido el 16-06-2019. URL: *https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274*.

— (2019). *Proof of Stake FAQ*. Accedido el 20-07-2019. URL: *https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ*.

Cantor Lyn; Cullen, Cam (2018). *The Global Internet Phenomena Report October 2018*. Accedido el 08-06-2019. URL: *https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf*.

Capella, Francisco (2009). *La banca con reserva fraccionaria*. Accedido el 08-06-2019. URL: *https://www.juandemariana.org/ijm-actualidad/analisis-diario/la-banca-con-reserva-fraccionaria*.

Casals, Josep (2016). *Learning to Fly with the Internet of Value*. Accedido el 15-06-2019. URL: *https://bankinnovation.net/2016/12/learning-to-fly-with-the-internet-of-value/*.

CME Open Markets - CME Group (2015). *What is an Internet of Value?* Accedido el 08-06-2019. URL: *http://openmarkets.cmegroup.com/10381/what-is-an-internet-of-value*.

Cochran, John P y Steven T Call (2000). «Free banking and credit creation: Implications for business cycle theory». En: *Quarterly Journal of Austrian Economics* 3.3, págs. 35-50. URL: *https://mises.org/library/free-banking-and-credit-creation-implications-business-cycle-theory-0*.

crypto-economy.net (2017). *¿Qué son las mining pools?* Accedido el 30-06-2019. URL: *https://www.crypto-economy.net/que-son-las-mining-pools/*.

CSBreakdown (2015a). *Elliptic Curve Cryptography and Diffie-Hellman*. Accedido el 23-06-2019. URL: *https://www.youtube.com/watch?v=yDXiDOJgxmg*.

— (2015b). *The Cryptography Behind Bitcoin*. Accedido el 23-06-2019. URL: *https://www.youtube.com/watch?v=5fSOd431l6A*.

CuriousInventor (2013). *Cómo funciona Bitcoin Under the Hood*. Accedido el 09-07-2019. URL: *https://www.youtube.com/watch?v=Lx9zgZCMqXE*.

Dai, Wei (1998). *B-Money*. Accedido el 08-06-2019. URL: *http://www.weidai.com/bmoney.txt*.

Dakota Quint (s.f.). *What is actually Ropsten? What is a new ânetworkâ?* Accedido el 10-11-2019. URL: *https://ethereum.stackexchange.com/a/13536*.

Darko, Emmanuel (2019). *What Is Blockchain 4.0?* Accedido el 21-07-2019. URL: *http://www.toks.tech/what-is-blockchain-4-0/*.

Davis, Joshua (2011). *The Crypto-Currency: Bitcoin and its mysterious inventor*. Accedido el 08-06-2019. URL: *https://www.newyorker.com/magazine/2011/10/10/the-crypto-currency*.

Dolores Fuentes (2016). *Tipos de criptografía: criptografía simétrica, criptografía asimétrica y criptografía híbrida.* Accedido el 22-06-2019. URL: *https://es.paperblog.com/tipos-de-criptografia-simetrica-asimetrica-y-hibrida-3713787/*.

Economist, T (2015). «The Great Chain of Being Sure about Things». En: *The Economist. Accedido desde http://www. economist. com*.

En areatecnologia (s.f.[a]). *COMO FUNCIONA INTERNET*. Accedido el 08-06-2019. URL: *https://www.areatecnologia.com/informatica/como-funciona-internet.html*.

— (s.f.[b]). *QUE ES UN SERVIDOR Y TIPOS DE SERVIDORES*. Accedido el 08-06-2019. URL: *https://www.areatecnologia.com/informatica/servidor-y-tipos.html*.

En circuito.io (s.f.). *circuito.io*. Accedido el 16-11-2019. URL: *https://www.circuito.io/*.

En DRAE (2018a). *Consenso Real Academia Española. (2018). Diccionario de la lengua española*. Accedido el 20-06-2019. URL: *https://dle.rae.es/?id=AP0O6TO*.

— (2018b). *Criptografía Real Academia Española. (2018). Diccionario de la lengua española*. Accedido el 22-06-2019. URL: *https://dle.rae.es/?id=BHcfHjo*.

En Espressif Systems (s.f.). *ESP32*. Accedido el 10-11-2019. URL: *https://www.espressif.com/en/products/hardware/esp32/overview*.

En Lexico (s.f.). *Defición de confianza*. Accedido el 15-06-2019. URL: *https://www.lexico.com/es/definicion/confianza*.

En Metamask (s.f.). *Brings Ethereum to your browser*. Accedido el 11-11-2019. URL: *https://metamask.io*.

En sparkfun.com (s.f.). *DHT22 Datasheet*. Accedido el 16-11-2019. URL: *https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf*.

En Wikipedia (s.f.[a]). *Atom (text editor)*. Accedido el 11-11-2019. URL: *https://en.wikipedia.org/wiki/Atom_(text_editor)*.

— (s.f.[b]). *Automatización industrial*. Accedido el 28-07-2019. URL: *https://es.wikipedia.org/wiki/Automatizaci%C3%B3n_industrial*.

— (s.f.[c]). *Blockchain: Structure*. Accedido el 08-06-2019. URL: *https://en.wikipedia.org/wiki/Blockchain#Structure*.

— (s.f.[d]). *Cadena de bloques*. Accedido el 16-06-2019. URL: *https://es.wikipedia.org/wiki/Cadena_de_bloques*.

— (s.f.[e]). *Contabilidad*. Accedido el 08-06-2019. URL: *https://es.wikipedia.org/wiki/Contabilidad#Historia*.

— (s.f.[f]). *Controlador lógico programable*. Accedido el 28-07-2019. URL: *https://es.wikipedia.org/wiki/Controlador_l%C3%B3gico_programable*.

— (s.f.[g]). *Criptografía*. Accedido el 22-06-2019. URL: *https://es.wikipedia.org/wiki/Criptograf%C3%ADa*.

— (s.f.[h]). *Criptografía Asimétrica*. Accedido el 22-06-2019. URL: *https://es.wikipedia.org/wiki/Criptograf%C3%ADa_asim%C3%A9trica*.

— (s.f.[i]). *Criptografía de curva elíptica*. Accedido el 23-06-2019. URL: *https://es.wikipedia.org/wiki/Criptograf%C3%ADa_de_curva_el%C3%ADptica*.

— (s.f.[j]). *Criptografía Simétrica*. Accedido el 22-06-2019. URL: *https://es.wikipedia.org/wiki/Criptograf%C3%ADa_sim%C3%A9trica*.

— (s.f.[k]). *Criptomoneda*. Accedido el 20-07-2019. URL: *https://es.wikipedia.org/wiki/Criptomoneda*.

— (s.f.[l]). *Doble Gasto*. Accedido el 20-06-2019. URL: *https://es.wikipedia.org/wiki/Doble_gasto*.

— (s.f.[m]). *Elliptic curve cryptography*. Accedido el 23-06-2019. URL: *https://en.wikipedia.org/wiki/Elliptic-curve_cryptography*.

— (s.f.[n]). *Ethash*. Accedido el 22-06-2019. URL: *https://en.wikipedia.org/wiki/Ethash*.

— (s.f.[o]). *Función hash*. Accedido el 22-06-2019. URL: *https://es.wikipedia.org/wiki/Funci%C3%B3n_hash*.

— (s.f.[p]). *Logaritmo discreto*. Accedido el 23-06-2019. URL: *https://es.wikipedia.org/wiki/Logaritmo_discreto*.

— (s.f.[q]). *Node.js*. Accedido el 11-11-2019. URL: *https://es.wikipedia.org/wiki/Node.js*.

— (s.f.[r]). *Nodo (informática)*. Accedido el 08-06-2019. URL: *https://es.wikipedia.org/wiki/Nodo_(inform%C3%A1tica)*.

— (s.f.[s]). *Privacy and blockchain*. Accedido el 08-06-2019. URL: *https://en.wikipedia.org/wiki/Privacy_and_blockchain*.

— (s.f.[t]). *Regulación automática*. Accedido el 24-07-2019. URL: *https://es.wikipedia.org/wiki/Regulaci%C3%B3n_autom%C3%A1tica*.

— (s.f.[u]). *SCADA*. Accedido el 28-07-2019. URL: *https://es.wikipedia.org/wiki/SCADA*.

— (s.f.[v]). *SHA-2*. Accedido el 22-06-2019. URL: *https://es.wikipedia.org/wiki/SHA-2*.

— (s.f.[w]). *Solidity*. Accedido el 11-11-2019. URL: *https://en.wikipedia.org/wiki/Solidity*.

Ezpeleta, J y P Álvarez (2018). *Lección 12 Algoritmos de consenso, Programación de Sistemas Concurrentes y Distribuidos*.

Fantom Foundation (2018). *An Introduction to DAGs and How They Differ From Blockchains*. Accedido el 21-06-2019. URL: *https://medium.com/fantomfoundation/an-introduction-to-dags-and-how-they-differ-from-blockchains-a6f703462090*.

Felix Lange (s.f.). *JSON RPC API*. Accedido el 10-11-2019. URL: *https://github.com/ethereum/wiki/wiki/JSON-RPC#json-rpc-api*.

Fercufer (2011). *Una Función hash en funcionamiento*. Accedido el 22-06-2019. URL: *https://commons.wikimedia.org/wiki/File:Hash_function2-es.svg*.

Gabriel Montes (2017). *Bitcoin Esencial Claves sobre claves privadas, públicas y direcciones*. Accedido el 24-06-2019. URL: *https://medium.com/@gab_montes/bitcoin-esencial-claves-sobre-claves-privadas-p%C3%BAblicas-y-direcciones-148f854b822d*.

Gill, Navdeep Singh (2018). *Building Decentralized Applications on BlockChain Technology*. Accedido el 15-06-2019. URL: *https://www.xenonstack.com/blog/decentralized-applications/*.

Glickstein, Bob (2019). *Understanding the Stellar Consensus Protocol*. Accedido el 20-07-2019. URL: *https://medium.com/interstellar/understanding-the-stellar-consensus-protocol-423409aad32e*.

Golin, Jonathan y Philippe Delhaise (2013). *The bank credit analysis handbook: a guide for analysts, bankers and investors*. John Wiley & Sons.

Govender, Seshree (2019). *Smart Contracts versus traditional contracts: same but different*. Accedido el 15-06-2019. URL: *https://ventureburn.com/2019/04/smart-contracts-vs-traditional-contracts/*.

Haber, Stuart y W Scott Stornetta (1990). «How to time-stamp a digital document». En: *Conference on the Theory and Application of Cryptography*. Springer, págs. 437-455.

hashgains (2018). *Bitcoin Vs Paypal*. Accedido el 15-06-2019. URL: *https://www.hashgains.com/comparison/bitcoin-vs-paypal?lang=es*.

Hertig, Alyssa (2017a). *How Do Ethereum Smart Contracts Work?* Accedido el 20-07-2019. URL: *https://www.coindesk.com/information/ethereum-smart-contracts-work*.

— (2017b). *How Ethereum Works*. Accedido el 20-07-2019. URL: *https://www.coindesk.com/information/how-ethereum-works*.

imchris (2004). *Cypherpunks Mailing List Information*. Accedido el 08-06-2019. URL: *https://web.archive.org/web/20160305051810/http://imchris.org/projects/cpunk.html*.

InterValue (2018). *Knowing the Blockchain 1.0, Blockchain 2.0, Blockchain 3.0, and Blockchain 4.0*. Accedido el 20-07-2019. URL: *https://medium.com/@intervalueproject/todays-blockchain-technology-has-undergone-several-iterations-50e7a0e037e3*.

iota community (2018). *What is IOTA?* Accedido el 28-07-2019. URL: *https://www.iota.org/get-started/faqs*.

Ivan (2017). *Aprende cómo funciona SSH y sus diferentes cifrados*. Accedido el 22-06-2019. URL: *https://www.laudemmedia.com/secure-shell-aprende-como-funciona-ssh-y-sus-diferentes-cifrados/*.

Jimi S. (2018). *Blockchain how a 51% attack works (double spend attack)*. Accedido el 21-06-2019. URL: *https://medium.com/coinmonks/what-is-a-51-attack-or-double-spend-attack-aa108db63474*.

Joshi, Archana Prashanth, Meng Han y Yan Wang (2018). «A survey on security and privacy issues of blockchain technology». En: *Mathematical Foundations of Computing* 1.2, págs. 121-147.

Khan, Ameer Tamoor y col. (2018). «Blockchain Technology with Applications to Distributed Control and Cooperative Robotics: A Survey». En:

Koblitz, Neal (1987). «Elliptic curve cryptosystems». En: *Mathematics of computation* 48.177, págs. 203-209.

kopanitsa (s.f.). *web3-arduino*. Accedido el 16-11-2019. URL: *https://github.com/kopanitsa/web3-arduino*.

Kristen Silverberg Conan French, Dennis Ferenzy (2016). *Getting Smart: Contracts on the Blockchain*. Accedido el 21-07-2019. URL: *https://www.iif.com/Publications/ID/582/Getting-Smart-Contracts-on-the-Blockchain*.

Lansiti Marco; Lakhani, Karim R. (2017). *The Truth About Blockchain*. Accedido el 08-06-2019. URL: *https://hbr.org/2017/01/the-truth-about-blockchain*.

learnmeabitcoin (s.f.). *TXID*. Accedido el 30-06-2019. URL: *https://learnmeabitcoin.com/glossary/txid*.

Lisk Academy (2019). *Consensus Protocols*. Accedido el 21-06-2019. URL: *https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/consensus-protocols*.

Manne, Robert (2011). *The Cypherpunk Revolutionary - Julian Assange*. Accedido el 08-06-2019. URL: *https://www.themonthly.com.au/issue/2011/february/1324596189/robert-manne/cypherpunk-revolutionary*.

Mao, Mingyang y Hong Xiao (2018). «Blockchain-based Technology for Industrial Control System CyperSecurity». En: *2018 International Conference on Network, Communication, Computer Engineering (NCCE 2018)*. Atlantis Press.

Mateo Cruz (2019). *Sistema de control: lazo abierto versus lazo cerrado*. Accedido el 24-07-2019. URL: *https://makinandovelez.wordpress.com/2019/04/03/sistema-de-control-lazo-abierto-versus-lazo-cerrado/*.

Max Thake (2018). *Blockchain vs. DAG Technology*. Accedido el 21-06-2019. URL: *https://medium.com/nakamo-to/blockchain-vs-dag-technology-1a406e6c6242*.

Miller, Victor S (1985). «Use of elliptic curves in cryptography». En: *Conference on the theory and application of cryptographic techniques*. Springer, págs. 417-426.

Moskov, Phillip (2018). *What Is Bit Gold? The Brainchild of Blockchain Pioneer Nick Szabo*. Accedido el 08-06-2019. URL: *https://coincentral.com/what-is-bit-gold-the-brainchild-of-blockchain-pioneer-nick-szabo/*.

Nakamoto, Satoshi y col. (2008). «Bitcoin: A peer-to-peer electronic cash system». En: URL: *https://bitcoin.org/bitcoin.pdf*.

nakamotoinstitute (2004). *RPOW - Reusable Proofs of Work*. Accedido el 08-06-2019. URL: *https://nakamotoinstitute.org/finney/rpow/*.

Narayan Prusty (2017). *Introduction to truffle*. Accedido el 11-11-2019. URL: *https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781787122147/8/ch08lvl1sec75/introduction-to-truffle*.

Narayanan, Arvind (2013). «What happened to the crypto dream?, part 1». En: *IEEE security & privacy* 11.2, págs. 75-76.

Narayanan, Arvind y col. (2016). *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. Princeton University Press.

Nick Adams, nadamsoreilly (2017). *Deterministic wallet*. Accedido el 24-06-2019. URL: *https://github.com/bitcoinbook/bitcoinbook/blob/develop/images/mbc2_0502.png*.

Nick Sullivan (2013). *A (relatively easy to understand) primer on elliptic curve cryptography*. Accedido el 23-06-2019. URL: *https://arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/*.

Official, genEOS (2018). *What Is Blockchain 4.0?* Accedido el 21-07-2019. URL: *https://medium.com/geneos/what-is-blockchain-4-0-613673875255*.

oroyfinanzas (2015). *¿Qué es el bloque Génesis en Bitcoin? Genesis block*. Accedido el 08-06-2019. URL: *https://www.oroyfinanzas.com/2015/01/que-es-bloque-genesis-bitcoin/*.

Otero, Cesar (2018). *Hackeo masivo de Quora: Roban 100 millones de cuentas del Yahoo respuestas inglés*. Accedido el 15-06-2019. URL: *https://as.com/meristation/2018/12/04/betech/1543928574_676934.html*.

Pastorino, Cecilia (2018). *Blockchain: qué es, cómo funciona y cómo se está usando en el mercado*. Accedido el 16-06-2019. URL: *https://www.welivesecurity.com/la-es/2018/09/04/blockchain-que-es-como-funciona-y-como-se-esta-usando-en-el-mercado/*.

Pease Lamport, Shostak (1995). *The byzantine generals problem, Advances in Ultra-Dependable Distributed Systems, N. Suri, CJ Walter, and M. M. Hugue*.

Pérez Solà, Cristina y Jordi Herrera Joancomartí (2014). «Bitcoins y el problema de los generales bizantinos». En:

Pérez, Miguel Ángel (2019). *Tipos de redes que hay para desarrollar proyectos con Blockchain: abiertas, federadas o privadas.* Accedido el 15-06-2019. URL: *https://icommunity.io/tipos-de-redes-que-hay-para-desarrollar-proyectos-con-blockchain-abiertas-federadas-o-privadas-by-miguel-angel-perez-icommunity-labs/*.

PetriB (2018). *The untold history of Bitcoin: Enter the Cypherpunks*. Accedido el 08-06-2019. URL: *https://medium.com/swlh/the-untold-history-of-bitcoin-enter-the-cypherpunks-f764dee962a1*.

Physicsch (2015). *PID Compensation Animated*. Accedido el 24-07-2019. URL: *https://en.wikipedia.org/wiki/File:PID_Compensation_Animated.gif*.

Preukschat, Alex (2017). *Los tipos de Blockchain: públicas, privadas e híbridas*. Accedido el 08-06-2019. URL: *https://www.iecisa.com/es/blog/Post/Los-tipos-de-Blockchain-publicas-privadas-e-hibridas-y-II/*.

Rautopia (s.f.). *Tipos de redes*. Accedido el 08-06-2019. URL: *https://commons.wikimedia.org/wiki/File:Centralizado-descentralizado-distribu%5C%C3%5C%ADdo.png*.

Ray, Shaan (2017). *What is Proof of Stake?* Accedido el 20-07-2019. URL: *https://hackernoon.com/what-is-proof-of-stake-8e0433018256*.

readthedocs.io (s.f.). *Gestión de Bases de Datos*. Accedido el 16-06-2019. URL: *https://gestionbasesdatos.readthedocs.io/es/latest/Tema1/Teoria.html*.

Recursos (2018). *Beneficios del sistema de control distribuido para el sector alimentación*. Accedido el 28-07-2019. URL: *https://tecnologiaparalaindustria.com/beneficios-del-sistema-de-control-distribuido-para-el-sector-alimentacion/*.

Renders, Matthijs (2018). *A Simple Guide to Understanding the Stellar Blockchain Network*. Accedido el 20-07-2019. URL: *https://hackernoon.com/a-simple-guide-to-understanding-the-stellar-blockchain-network-3609d728e9a7*.

Romero, Cristina (2007). *El riesgo de las TICs. Tecnologias de la información y la comunicación*. Accedido el 08-06-2019. URL: *http://elfisco.com/articulos/el-riesgo-de-tics-tecnologia-de-la-informacion-y-la-comunicacion/*.

Rosa, Fidel La (2018). *Blockchain y bases de datos descentralizadas ¿son la misma cosa?* Accedido el 16-06-2019. URL: *https://www.criptonoticias.com/redes-protocolos/blockchain-y-bases-de-datos-descentralizadas-son-la-misma-cosa/*.

Rosic, Ameer (2016). *What is Blockchain Technology? A Step-by-Step Guide For Beginners*. Accedido el 28-07-2019. URL: *https://blockgeeks.com/guides/what-is-blockchain-technology/*.

— (2017). *Proof of Work vs Proof of Stake: Basic Mining Guide*. Accedido el 20-07-2019. URL: *https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/*.

— (2018). *Basic Primer: Blockchain Consensus Protocol*. Accedido el 20-06-2019. URL: *https://blockgeeks.com/guides/blockchain-consensus/*.

RSK educate (2015). *RSK educate: Executive Course Blockchain and Smart contracts powered by RSK*. Accedido el 08-06-2019. URL: *https://www.rsk.co/*.

Schumann, Turner (2018). *Consensus Mechanisms Explained: PoW vs. PoS*. Accedido el 20-07-2019. URL: *https://hackernoon.com/consensus-mechanisms-explained-pow-vs-pos-89951c66ae10*.

sheinix (2018). *Transacciones en Bitcoin*. Accedido el 30-06-2019. URL: *https://medium.com/@sheinix/transacciones-en-bitcoin-841a087ff439*.

Shirriff, Ken (2014). *Bitcoins the hard way Using the raw Bitcoin protocol*. Accedido el 30-06-2019. URL: *http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html#ref13*.

Stellar Development Foundation (2015). *On Worldwide Consensus The future is distributed a summary of the Stellar Consensus Protocol white paper*. Accedido el 21-06-2019. URL: *https://medium.com/stellar-development-foundation/on-worldwide-consensus-359e9eb3e949*.

Tabora, Vince (2018). *Databases and Blockchains, The Difference Is In Their Purpose And Design*. Accedido el 16-06-2019. URL: *https://hackernoon.com/databases-and-blockchains-the-difference-is-in-their-purpose-and-design-56ba6335778b*.

Tactical Technology Collective y Front Line (2018). *MANTENER TUS COMUNICACIONES DIGITALES PRIVADAS*. Accedido el 08-06-2019. URL: *https://securityinabox.org/es/guide/secure-communication/*.

Tim Coulter (s.f.). *USING INFURA (OR A CUSTOM PROVIDER)*. Accedido el 11-11-2019. URL: *https://www.trufflesuite.com/tutorials/using-infura-custom-provider*.

Toledo, Guillermo (2018). *Estructura de datos en la blockchain*. Accedido el 30-06-2019. URL: *https://social-blockchain.org/2018/12/04/estructura-de-datos-en-la-blockchain/*.

Unibright.io (2017). *Blockchain evolution: from 1.0 to 4.0*. Accedido el 20-07-2019. URL: *https://medium.com/@UnibrightIO/blockchain-evolution-from-1-0-to-4-0-3fbdbccfc666*.

Vigna, Paul y Michael J Casey (2019). *The Truth Machine: The Blockchain and the Future of Everything*. Picador.

Villagómez, Carlos (2017). *Introducción a las bases de datos*. Accedido el 16-06-2019. URL: *https://es.ccm.net/contents/66-introduccion-a-las-bases-de-datos*.

W, Oscar (2018). *AI Smart Contracts, Past, Present, and Future*. Accedido el 15-06-2019. URL: *https://hackernoon.com/ai-smart-contracts-the-past-present-and-future-625d3416807b*.

Wander, Matthaus (2013). *Bitcoin Block Data*. Accedido el 30-06-2019. URL: *https://commons.wikimedia.org/wiki/File:Bitcoin_Block_Data.png*.

Wang, Ye Diana y Henry H Emurian (2005). «An overview of online trust: Concepts, elements, and implications». En: *Computers in human behavior* 21.1, págs. 105-125.

Winterfield, Andrew (2018). *Consensus algorithms pos dpos*. Accedido el 20-07-2019. URL: *https://en.bitcoinwiki.org/index.php?curid=274945*.

Xavier Decuyper, Simply Explained Savjee (2017). *Encriptación asimétrica: simplemente explicada*. Accedido el 22-06-2019. URL: *https://www.youtube.com/watch?v=AQDCe585Lnc*.